AD-A154 874

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NSWC TR 82-171	2. GOVT ACCESSION NO. AD-A154 874	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A SURVEY OF SOFTWARE RELIABILITY MODELING AND ESTIMATION		5. TYPE OF REPORT & PERIOD COVERED Final
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Dr. William H. Farr		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Surface Weapons Center (Code K52) Dahlgren, Va 22448		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS F66312 62766N
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Surface Weapons Center Dahlgren, VA 22448		12. REPORT DATE September 1983
		13. NUMBER OF PAGES 172
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Software reliability, Software Reliability Models, software error rate, Mean Time Before Failure, Poisson process, Markov process, estimation, software testing, Error Seeding/Tagging, Data Domain, Bayesian Software Models

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

With the ever-increasing role that software is playing in the weapon systems, a great need has arisen for tools that are useful in developing cost-effective software. An area of research has arisen over the last 10 years in providing a software manager quantitative statements about the reliability of the software. Using this quantitative measure, the manager

(Continue on the other side)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. (Cont'd)

can make a determination of when software testing should terminate and how to best utilize testing personnel. This report discusses the various approaches that have been advocated for reliability estimation. It reviews the various models that have been proposed for this estimation process, providing the model assumptions, the estimates of reliability, the precision of those estimates, and the data required for their implementation. A comparison is then made among some of these models based upon studies that have been done. General comments concerning software reliability implementation are discussed in the final section of the report.

FOREWORD

This work was completed while the author was a member of the Submarine Launched Ballistic Missile (SLBM) Software Development Division (K50) of the Naval Surface Weapons Center (NSWC), Dahlgren, Virginia. The work was supported under Independent Exploratory Development (IED) funding, project number F66312.

The author wishes to thank the members of the IED Committee for allowing him to explore this research area. The author would also like to thank Mrs. Doreen Daniels, who at the time of this research was the Head of the Quality Assurance Branch (K52), for allowing the author the opportunity to work on this project.

Approved by:

O. F. Braxton

O. F. BRAXTON, Head
Strategic Systems Department

DTIC
ELECTE
JUN 10 1985
S D
B

Accession For
NTIS GRA&I ☑
DTIC TAB ☐
Unannounced ☐
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

CONTENTS

Chapter			Page
1	INTRODUCTION		1-1
2	ERROR SEEDING/TAGGING MODELS		2-1
3	DATA DOMAIN APPROACH		3-1
4	TIME DOMAIN APPROACH		4-1
	4.1	HARDWARE VERSUS SOFTWARE RELIABILITY MODELING	4-2
	4.2	CLASSICAL SOFTWARE MODELS	4-3
	4.2.1	Weibull Model	4-3
	4.2.2	Shooman Model	4-7
	4.2.3	Jelinski and Moranda "De-Eutrophication" Model	4-11
	4.2.4	Schick-Wolverton Model	4-33
	4.2.5	Generalized Poisson Model	4-41
	4.2.6	Geometric Model	4-51
	4.2.7	Geometric Poisson Model	4-63
	4.2.8	Schneidewind's Model	4-66
	4.2.9	Nonhomogeneous Poisson Process	4-71
	4.2.10	Duane's Model	4-80
	4.2.11	Execution Time Model	4-82
	4.2.12	Brooks and Motley's Models	4-93
	4.3	BAYESIAN MODELS	4-100
	4.3.1	Littlewood's Bayesian Debugging Model	4-101
	4.3.2	Littlewood and Verrall's Bayesian Reliability Growth Model	4-108
	4.3.3	Thompson and Chelson's Bayesian Reliability Model	4-112
	4.4	MARKOV MODELS	4-116
	4.4.1	Trivedi and Shooman's Many State Markov Models	4-116
	4.4.2	Littlewood's Semi-Markov Model	4-120
5	COMPARISON OF RELIABILITY MODELS		5-1
6	"QUICK" ESTIMATES OF SOFTWARE RELIABILITY MEASURES		6-1
	6.1	MTBF ESTIMATION	6-1
	6.2	PRAGMATIC SOFTWARE RELIABILITY ESTIMATION	6-2

CONTENTS (Cont'd)

Chapter		Page
7	SUMMARY AND CONCLUSIONS	7-1
REFERENCES	. . . . .	8-1
BIBLIOGRAPHY	. . . . .	9-1
DISTRIBUTION	. . . . .	(1)

ILLUSTRATIONS

Figure		
3-1	THE INPUT SPACE $E = E_u$ (user space)$\bigcup \bar{E}_u$	3-7
4-1	DE-EUTROPHICATION PROCESS	4-12
4-2	SCHICK-WOLVERTON HAZARD RATE FUNCTION	4-34
4-3	GEOMETRIC "DE-EUTROPHICATION" PROCESS	4-52
4-4	MODIFIED GEOMETRIC "DE-EUTROPHICATION" PROCESS	4-60

CHAPTER 1

INTRODUCTION

Since the early 1970's, tremendous growth has been seen in the development of computer software for weapon systems. Part of this development is a result of the development of microprocessors and distributed processing and networking. With miniaturization of components in computer systems came the ever-increasing role of software in the weapon systems. The software is called upon to perform more complex tasks than ever before, e.g., weapon's coordination, scheduling, and control. This increased role has also meant a dramatic increase in software costs. In 1975, it was estimated that software costs exceeded hardware costs by a factor of three or four for U.S. Air Force weapons systems.[1] In 1977, the costs of software alone to the entire U.S. economy ranged from $10 to $19 billion.[2]

With such an increasing reliance on computer systems, there is a major problem in developing "error-free" programs. For large scale real-time embedded computer systems such as the TRIDENT-I Fire Control System (TFCS) and its follow-on, TRIDENT-II, it is an impossible task to check every conceivable logic path in the computer code for every combination of possible inputs, to discover "programming errors." Researchers and practitioners of software code development have looked to various tools to cut down on the number of errors in the design and development stage. Included among these tools or approaches are: structured code, a "top-down" approach to the software design, and the development of a number of automated verification and validation (V&V) tools for program checkout; however, they have not proven to be a complete answer. A quote from the July 1973 Air Force Magazine states:

> "The world's most carefully planned and generously funded software program was that developed for the Apollo series of lunar flights. The effort attracted some of the nation's best computer programmers and involved two competing teams. Checking the software as thorough as the experts knew how to make it. In the aggregate, about $600 million was spent on software for the Apollo program. Yet almost every major fault of the Apollo program, from false alarms to actual mishaps, was the direct result of errors in computer software."[3]

Another problem that the U.S. Government and the Department of Defense (DOD), in particular, are facing with software procurement is the inability to establish and enforce software reliability goals from contractors. "How does one develop guidelines or standards that can be used to ensure a certain quality in the software as is currently imposed in military standards for hardware development?" A more basic question that needs to be answered first is: "What is meant by software reliability?" First, specific goals or objectives need to be established.

This report defines software reliability as "the probability that a given software program will operate without failure for a specified time in a specified environment." The specified environment is particularly emphasized as it constitutes one of the major assumptions for many of the reliability models discussed in this report. If the testing environment is quite a bit different from the actual operating environment, the program's reliability cannot be accounted for in that environment. Software error or failure is defined as "any occurrence attributable to software in which the system did not meet its performance requirements." These definitions are consistent with the majority of such definitions found in the literature.

Knowing the current status of the program reliability can determine when testing should be completed and the program released for operational use. It can also aid the software manager in determining how best to allocate his limited resources (manpower, computer time) among the various program modules for testing. The current program reliability can be used in making decisions regarding design tradeoffs between reliability, costs, performance, and schedule. Another use is in evaluating various software engineering approaches or tools to find the one that leads to the "most reliable" program with (hopefully) the minimum cost. The literature is sadly lacking on controlled studies which indicate the performance of software tools in eliminating errors in software code.

The purpose of this report is to provide a survey of the various approaches that have appeared in the literature concerning the estimation or modeling of a program's reliability. This report describes the underlying assumptions for each of the models and provides a data requirements list for implementation. The various models are contrasted with each other and the relative merits or drawbacks are also highlighted. This report provides a practical guide for the implementation of these procedures on a software program. Finally, the report gives any results of studies undertaken to analyze the performance of these procedures. Unfortunately, this is one of the areas in which little has been done. Most of these studies are either based upon simulated data or data sets for which the data were collected for purposes other than reliability modeling. As a result, some of the key assumptions upon which these models or approaches rest are violated. An additional purpose of this report is to provide the assumptions and data requirements for the various models. Steps will be taken in the software development for TRIDENT-II to ensure the compatibility of the data with the model assumptions.

Over the last 15 years, these models and estimation procedures have evolved. There are basically three different approaches that have been identified in the literature: Error Seeding/Tagging Models, the Data Domain Approach, and the Time Domain Approach. Chapter 2 of this report describes the Error Seeding/Tagging Models, Chapter 3 describes the Data Domain Approach, and Chapter 4 describes the Time Domain Modeling efforts. Chapter 5 describes any studies and their conclusions in comparing the performance of these various approaches on actual data sets. Finally, Chapter 6 presents a number of "quick" estimates of reliability.

Before beginning the description of these various approaches, it must be kept in mind throughout this report that software reliability modeling is just one of many tools. It cannot provide all of the answers that the software managers must

face. It must be taken as a bit of information, which along with others, is helpful in making a realistic judgement concerning a program's status. Because of the current controversy about which of the models is best and because of the uncertainty about the performance of the software reliability modeling approaches, it is emphasized that the model that is best suited to the data be applied. The resulting estimate of reliability may be used as another source of information in determining program status.

CHAPTER 2

ERROR SEEDING/TAGGING MODELS

This approach, first proposed by Mills,[4] involves "seeding" a given program with a number of known errors. The assumption is made that the distribution of the "seeded" errors is the same as the distribution for the inherent errors in the program. The program is then given over to a testing team for V&V. Some of the errors discovered by the testing team are seeded errors while others are inherent in the program. Using these counts, the total number of errors inherent in the program can be estimated. In particular, if there are N errors inherent in the program and n are randomly inserted with r errors being subsequently detected by the quality assurance (QA) team [k (k<r) being seeded errors], it can be shown that the maximum likelihood estimate (MLE) of N is:

$$\hat{N} = \left[\frac{n(r - k)}{k}\right] \quad , \tag{2.1}$$

with [ ] being the greatest integer function.

The biggest drawback to this Seeding Approach is the assumption that is made about the distribution of seeded errors being the same as the distribution of inherent errors. This is an impossible assumption to check, especially in the latter stages of program development. At that point, many of the easy errors (e.g., misspelled output) have been eliminated and the only remaining errors are the very subtle errors which are extremely difficult to uncover.

Another approach, proposed by Rudner,[5] avoids this problem by employing a "two-stage" or "two-team" testing procedure. The program is first given to one team for testing which finds n errors. The program is then turned over to a second testing team which discovers a total of r errors, (k of which were also found by the first team.) Using the hypergeometric distribution, the MLE for the total number of errors in the program, N, can be shown as:

$$\hat{\hat{N}} = \left[\frac{nr}{k}\right] \quad , \tag{2.2}$$

where again [ ] denotes the greatest integer function.

In an article by Schick and Wolverton,[6] reference is made to a pair of papers by Basin[7,8] in which the following approach is taken. Suppose a program consists of M statements from which n are randomly selected and errors are introduced. If

r statements are then randomly chosen and tested with k_1 having inherent errors and k_2 having seeded errors, then the MLE of N, the total number of errors, is:

$$\hat{\hat{\hat{N}}} = \left[k_1 \frac{(M - n + 1)}{r - k_2} \right] , \tag{2.3}$$

with [ ] being the greatest integer function.

All of these procedures stem from "capture/recapture" estimation techniques which estimate the total number of animals of a given species. A "tagged" set of animals is released into the environment and after allowing the animals sufficient time to disperse, a second capture is made. Based upon the number of tagged animals released and recaptured, estimates of the total population size can be made.

In applying these estimation procedures, Schick and Wolverton[6] warn that, based upon preliminary calculations, the tag ratio (the average number of tagged errors in the sample) should be greater than 20 to ensure the estimates are close to being unbiased. These estimation procedures can be applied at any point in the life cycle development of a program to estimate the current error content. The biggest drawbacks are in seeding the errors and in the employment of limited resources in a two-team approach. Few organizations can afford the luxury of duplicate testing teams for a given program or even program modules. Generally, in the life cycle development of a program, if schedules start to slip, the time is made up at the expense of the V&V effort. As a result, when the program reaches the testing team, all available resources are spent to quickly perform the testing tasks and release the program to the operational user. These procedures also do not provide time-dependent reliability measures of the software, which may or may not be a drawback. This is discussed in Chapter 4, the Time Domain Approach.

CHAPTER 3

DATA DOMAIN APPROACH

The Data Domain Approach includes those procedures that estimate a program's current reliability based strictly on the number of successful runs observed compared to the total number of runs made. Included within this category are procedures that try to employ test inputs for the program that are chosen according to probability distributions of anticipated operational usage. The various inputs to a program are broken up into categories, and probabilities are then assigned to those categories which represent anticipated uses.

For example, range might be an input. It can be broken up into the categories [0, 1500 nautical miles (nmi)], [1501 nmi, 2500 nmi], [2501 nmi, 3500 nmi] and [3501 nmi or more]. Probabilities are then assigned to each category, based upon the anticipated operational usage. If it is anticipated that about one-fourth of all the ranges are 3500 nmi or more, that category is assigned the probability ¼. The inputs are randomly selected according to their probability distributions and the resulting test cases are run. The estimated reliability is then simply the total number of successful runs over the total number of test runs.

The Data Domain procedures try to divest themselves of time between error occurrence that the models of the Time Domain Approach may employ. If time is a factor in some of these models, it represents the total elapsed time (either wall clock time or CPU time) for a testing session and not the times of error occurrence.

If a random selection of inputs, which reflect the anticipated operational use, is made and N runs are made, with S being successful, then the estimate of the current program reliability is:

$$\hat{R} = S/N. \tag{3.1}$$

Using this basic Binomial Experiment Approach, a number of researchers have proposed modifications of this estimator. Hecht[9] proposed the estimators:

$$\hat{R}_1 = (S/(NxL)) \tag{3.2}$$

and

$$\hat{R}_2 = (S/(NxLxW)) \tag{3.3}$$

where L is the number of machine instructions submitted and W is the average number of bits-per-instruction. The modified estimators allow for differences in exposure to failure (by normalizing the estimator by the program length) and for differences in programs operating on different machines with different word sizes. Suppose the reliability estimates between two programs are compared. If one program is on a large main-frame computer that takes a large amount of time to process and the other program is on a small microprocessor which executes quickly, the results may be misleading if the reliability is calculated using equation (3.1).

Brown and Lipow[10] have suggested a modification of the basic estimator to allow for the fact that many times in testing, the input is chosen to "stress" the software program. The resulting estimate of reliability then tends to be on the "pessimistic side." Their procedure is to take the input space and divide it into "homogeneous" regions, P_i, i=1,...,K. They are homogeneous in the sense of fault generation. Suppose N_j runs are made from the partition region P_j, and F_j are failures. The estimate of the "unreliability" of that region is $\frac{F_j}{N_j}$. If the probability in an operational environment of drawing points from P_j is $P\{P_j\}$, the unreliability for the entire input space can be estimated as the sum over all regions of the corresponding unreliability of that region times the probability of drawing an input point from that region, i.e.,

$$\text{Estimate of the unreliability for the program} = \sum_{i=1}^{K} \frac{F_i}{N_i} P\{P_i\}. \qquad (3.4)$$

The estimate of the reliability of the program is then given as 1 minus the unreliability. The main drawbacks of this approach are the construction of partition sets which are homogeneous with respect to error generation and assigning a probability that an input point be drawn from a given partition region. The former is impossible to determine while the latter introduces a lot of errors in the estimate based upon subjective judgement.

Corcoran, Weingarten, and Zehna[11] proposed a model which is more suited to hardware reliability applications, but because of its easy extension to software reliability modeling and the fact that it is a generalization of the previous binomial, it is mentioned here. Suppose there are M sources or types of software errors that can occur. And suppose a_i is the probability that if the ith type of error is observed, it is corrected, i.e., the conditional probability

$$P\{\text{error corrected} \mid \text{ith type observed}\} = a_i$$

where

$$i = 1,\ldots,M.$$

If N runs are made and F_i errors of the ith type are observed, then the estimate of the reliability of the program is:

$$\hat{R} = \frac{S}{N} + \sum_{i=1}^{M} y_i(F_i/N) \tag{3.5}$$

where S is the total number of successful runs and

$$y_i = \begin{cases} a_i & \text{if } F_i > 0 \\ 0 & \text{if } F_i = 0 \end{cases} . \tag{3.6}$$

This estimator can be shown to be asymptotically unbiased and its variance goes to zero for large N. One drawback for this model is that the M types of error sources have to be known beforehand, and the most serious drawback is knowing the a_i's.

These next models discussed try to combine not only the results of a given set of runs, as in the previous models, but they also try to take into account the input space.

The first model is one by Nelson.[12] The basic assumptions are:

Assumptions

(a) A program may be defined as a specification of a computable function F on a set E

$$E = (E_i : i=1,\ldots,N)$$

which is the set of all data input values needed to execute the program.

(b) Execution of the program for each input E_i produces output $F(E_i)$.

(c) Because of imperfections in the program, the program actually specifies a function F′ which differs from the intended function F.

(d) For some of the E_i, the actual output $F'(E_i)$ is within an acceptable tolerance of the intended output $F(E_i)$; i.e.,

$$\left|F'(E_i) - F(E_i)\right| \leq \Delta_i. \tag{3.7}$$

But for some E_j, the actual output $F'(E_j)$ is not within acceptable limits; i.e.,

$$\left|F'(E_j) - F(E_j)\right| > \Delta_j \tag{3.8}$$

and an error is said to occur. N may be very large, but it is finite, owing to the fact that only a finite number of different values can fit into the word size

of the computer. Now suppose that E_e is the set of all inputs producing errors on a given run; i.e.,

$$E_e = \{E_j: \left|F'(E_j) - F(E_j)\right| > \Delta_j\} \tag{3.9}$$

Suppose there are n_e elements in this set. Then the probability of the program executing correctly if an input is randomly selected from E is:

$$R = 1 - \frac{n_e}{N} . \tag{3.10}$$

However, the usual situation is that the points E_i are not chosen randomly. They are chosen according to some operational requirement which can be represented as a probability distribution over E. For this distribution,

$$p_i = P\{E_i \text{ is selected}\}.$$

Hence the reliability of the program can be expressed in terms of these probabilities as:

$$R = \sum_{i=1}^{N} p_i(1 - y_i) \tag{3.11}$$

where

$$y_i = \begin{cases} 0 & \text{if } E_i \notin E_e \\ 1 & \text{if } E_i \varepsilon E_e \end{cases} . \tag{3.12}$$

If n runs are made and the inputs are chosen according to the probability distribution over E, then the probability of all runs being successful is:

$$R_n = R^n = \left[\sum_{i=1}^{N} p_i(1 - y_i)\right]^n . \tag{3.13}$$

Nelson[12] expands his model by allowing for the fact that usually runs are not made independently of each other, i.e., one of the input variables may be chosen in ascending order from run-to-run. To allow for this, Nelson redefines the probability distribution over the input space as:

$$p_{ij} = P\{E_i \text{ is selected on the jth run of the sequence}\}. \tag{3.14}$$

Hence, for the jth run, the probability of a failure is:

$$p_j = \sum_{i=1}^{N} p_{ij}y_i, \quad \text{with } y_i \text{ as previously defined.} \tag{3.15}$$

The probability that there are no failures in n runs becomes:

$$R_n = (1 - p_1)\ (1 - p_2)\ldots(1 - p_n). \tag{3.16}$$

As in the previous binomial type models, the estimated reliability based on Nelson's first model is simply:

$$\hat{R} = 1 - \frac{f}{n}\ , \tag{3.17}$$

where f is the total number of failures and the input points are chosen according to the probability distribution over E. To construct this probability distribution, Nelson suggests that the ranges of the various input variables be broken up into subranges. Probabilities are then assigned to these subranges based upon anticipated operational usage.

In the TRW report by Thayer,[13] the model by Nelson is again modified by taking the input space E and partitioning it into disjoint regions, R_i i=1,...,k, i.e.,

$$E = \bigcup_{i=1}^{k} R_i \text{ and } R_i \cap R_j = \phi.$$

The probability that a point is randomly selected from R_j can be calculated as:

$$P_{R_j} = \sum_{E_i \varepsilon R_j} p_i\ ; \tag{3.18}$$

i.e., all the operational probabilities of the input points falling into region R_j are summed over. If the region R_j is further divided into two sets R_j' and R_j'' where

$$R_j' = \{E_i \varepsilon R_j \cap \bar{E}_e\} \tag{3.19}$$

and

$$R_j'' = \{E_i \varepsilon R_j \cap E_e\}, \tag{3.20}$$

the set R_j' is derived consisting of all input points in R_j, which result in successful execution of the program. R_j'' consists of all input points of R_j yielding failures.

The probability of a point falling in R_j' is therefore:

$$P_{R_j'} = \sum_{E_i \varepsilon R_j} (1 - y_i)p_i\ , \tag{3.21}$$

with $y_i = 0$ if $E_i \varepsilon R'_j$, and 1 otherwise. The probability of a point falling in R''_j is similarly computed as:

$$P_{R''_j} = \sum_{E_i \varepsilon R_j} y_i p_i \quad . \tag{3.22}$$

The overall reliability of the program can then be expressed in terms of these probabilities as:

$$R = 1 - \sum_{i=1}^{N} y_i p_i \quad , \tag{3.23}$$

$$= 1 - \sum_{j=1}^{K} \sum_{E_i \varepsilon R_j} y_i p_i \quad , \tag{3.24}$$

$$= 1 - \sum_{j=1}^{K} P_{R''_j} \quad , \tag{3.25}$$

and

$$= \sum_{j=1}^{K} P_{R'_j} \quad . \tag{3.26}$$

The input space has been stratified into regions in exactly the same manner as described by Brown and Lipow.[10] Brown and Lipow created their strata based upon creating regions which were homogeneous with respect to error generation, while Nelson[12] suggested a partitioning based upon logic paths.

Using results from basic sampling theory for a stratified population, if n_j runs are made in region R_j, and f_j are failures, then the estimate of R is again simply:

$$\hat{R} = 1 - \sum_{j=1}^{K} \frac{f_j}{n_j} P_{R_j}, \tag{3.27}$$

provided the input points in R_j are chosen according to the probability distribution over the input space E. The variance of this estimator can be shown to be

$$\text{Var}\ \{\hat{R}\} = \sum_{j=1}^{K} \frac{P_{R'_j}\ P_{R''_j}}{n_j} \quad . \tag{3.28}$$

This variance can be minimized by taking

$$n_j \cong n\ \frac{\sqrt{P_{R'_j}\ P_{R''_j}}}{\sum_{i=1}^{K}\sqrt{P_{R'_i}\ P_{R''_i}}} \quad . \tag{3.29}$$

As in the previous Nelson models, the biggest drawback is the establishment of the distribution over the input space to determine $P_{R'_j}$ and $P_{R''_j}$.

A paper by Sugiura, Yamamoto, and Shiino[14] also considers the binomial model, but views the input space as being composed of two parts. The input space is broken up into a user space, the space where the input is actually drawn from, and its complement (Figure 3-1).

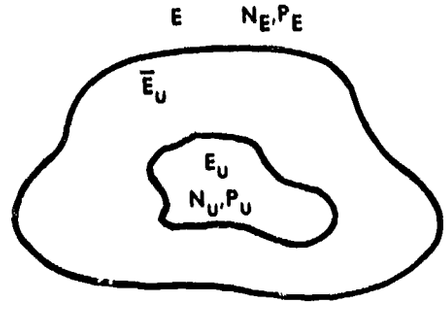

FIGURE 3-1. THE INPUT SPACE E = E_u (USER SPACE) $\cup$ $\bar{E}_u$.

The area $\bar{E}_u$ is never tested in the V&V stage. For the entire input space E, suppose there are a total of N_E input points, of which a proportion P_E of them result in errors. The real reliability of the program is therefore:

$$R_E = 1 - P_E \quad . \tag{3.30}$$

However test inputs are obtained and sampled from the user space E_u, where there are a total of N_u points, of which a proportion P_u result in errors.

Now suppose a set of n runs are made and f result in failures. P_u can be estimated, as seen before, as

$$P_u \simeq \frac{f}{n} \quad \text{for n large.} \tag{3.31}$$

Now suppose that m bugs are eliminated by debugging. Then the following expression is derived

$$P_u^{j-m} = P_u^j - \frac{m}{N_u} \tag{3.32}$$

where P_u^j is the failure probability before debugging and P_u^{j-m} is the failure probability after debugging. The points P_u^j and P_u^{j-m} are estimated using equation (3.31) on two separate testing occasions. Equation (3.32) can be plotted as a function of m by holding P_u^j fixed. Calculating the slope of this line, N_u can be estimated as

$$\hat{N}_u = -\frac{1}{\text{Slope}} . \tag{3.33}$$

Least squares can also be used to fit equation (3.32) to data. The software becomes perfect when

$$P_u^{j-m} = P_u^j - \frac{m}{N_u} = 0 ; \tag{3.34}$$

i.e., when

$$m = P_u^j N_u , \tag{3.35}$$

which simply is when the number of errors removed is the same as the total number of input points leading to failures. This assumes no new errors are introduced in the debugging process. If there are bugs which have complex causes, additional software errors might be introduced when correcting those errors. Suppose that such bugs exist in equal probabilities in the input space. Then the probability of such bugs residing in the user space is N_u/N_E. If such bugs in their correction produce B new errors, the failure probability, after the elimination of m errors, is:

$$P_u^{j-m} = P_u^j - \frac{m}{N_u}\left(1 - \frac{N_u}{N_E} B\right). \tag{3.36}$$

The program is error free when

$$P_u^{j-m} = P_u^j - \frac{m}{N_u}(1 - D) = 0 \tag{3.37}$$

i.e.,

$$m = \frac{P_u^j N_u}{1 - D} \tag{3.38}$$

where

$$D = \frac{N_u B}{N_E}. \tag{3.39}$$

If the program is at a particular point in the testing stage where the failure probability is P_u^j

then

$$m = \frac{P_u^j N_u}{1 - D} \tag{3.40}$$

is called the "remaining bug index." The unknowns in equation (3.36) can be estimated, using the results of several testing sessions and equation (3.31).

This is a very simplistic model that has not been employed on any data sets. Their formulation also rests heavily on obtaining good estimates of the P_u^j's using equation (3.31) in order to fit the straight line as a function of m. This means that the number of runs for a given testing session should be quite large.

The next reference in the Data Domain section is to a paper by Elliot, et. al.[15] The techniques in their paper again employ the simplistic assumption of a binomial experiment. For this reason, they assume that the runs are independent and the true reliability is the percentage of points in the input space which result in the program running correctly, i.e.,

$$R = \frac{S}{N} \quad , \tag{3.41}$$

where S is the total number of points in the input space E, which result in successful execution of the program. N is the total number of points in that space.

They propose two testing procedures to determine whether a program has reached a given reliability. One is based on a fixed sample size, the other is a sequential testing procedure. The fixed sample size is the usual hypothesis testing procedure for a binomial probability. The user specifies a size for the Type I error (the probability of rejecting the program when it has a desired reliability level) and a Type II error (the probability of accepting a program when the reliability is no more than a specified level). Using these values, tables are given which provide the number of tests, n, to run and the maximum number of failures, f, that are allowed if the program is to be accepted. The program tester randomly selects a subset of n input points from the input space E and runs the program. If more than f program failures are observed, the hypothesis that the program has reached the desired reliability level is rejected; otherwise, it is accepted.

For the sequential procedure, the tester specifies: a minimum acceptable reliability R_{min}, a probability α that the program with this reliability will pass testing, a probability R_{max} for which one wants to be "almost sure" that the software will pass, and a probability β that the software with this probability will fail the test. The sequential procedure is to:

(a) Accept the software if

$$F \leq -h_2 + BN_T \quad ;$$

(b) Reject the software if

$$F \geq h_1 + BN_T \quad ;$$

(c) Otherwise, continue testing

where F is the total number of failures experienced up through N_T tests (N_T=1, 2,...),

$$h_1 = [\ell n(1-\alpha) - \ell n\beta]/D \quad , \tag{3.42}$$

$$h_2 = [\ell n(1-\beta) - \ell n\alpha]/D \quad , \tag{3.43}$$

$$B = [\ell nR_{max} - \ell nR_{min}]/D \quad , \tag{3.44}$$

and

$$D = \ell nR_{max} - \ell nR_{min} - \ell n(1 - R_{max}) + \ell n(1 - R_{min}) \tag{3.45}$$

As an example, suppose a program is tested in which if the true reliability is something less than $R_{min} = .7$ it is desirable to limit the risk of releasing it by setting $\alpha = .05$. On the other hand, if the program has a reliability of .95 or larger, the chance of rejecting it should be $\beta = .1$.

Then

$$h_1 = [\ell n.95 - \ell n.1]/D \quad , \tag{3.46}$$

$$h_2 = [\ell n.9 - \ell n.05]/D \quad , \tag{3.47}$$

$$B = [\ell n.95 - \ell n.7]/D \quad , \tag{3.48}$$

and

$$D = \ell n.95 - \ell n.7 - \ell n(1-.95) + \ell n(1-.7). \tag{3.49}$$

Performing the calculations, it is found that:

$$D = 2.097 \quad , \tag{3.50}$$

$$h_1 = 1.074 \quad , \tag{3.51}$$

$$h_2 = 1.378 \quad , \tag{3.52}$$

and

$$B = .146 \quad . \tag{3.53}$$

So the sequential procedure is:

(a) Accept the software if:

$$F \leq -1.378 + .146N_T;$$

(b) Reject the software if:

$$F \geq 1.074 + .146N_T;$$

(c) Otherwise, continue testing.

The advantage of the sequential procedure over the fixed sample size is that on the average, a sequential procedure requires less testing than a fixed sample size to achieve the same levels for the Type I and Type II errors.

The last model considered in this section is LaPadula's Reliability Growth Model[16] (see also References 17 and 18). The approach is to fit, using least squares, a reliability curve through the success/failure counts observed at various stages of the software testing. More specifically, the assumptions are:

Model Assumptions

(a) Testing is conducted in a series of N stages. A stage is marked by any change or modification to the program.

(b) At each stage, n_i i=1,...,N tests are performed of which S_i are successful. The number of tests performed at a given stage is not fixed in advance.

(c) After the completion of the N-th stage (which itself is not set in advance), a growth curve of the form

$$R(k) = R(u) - A/k \tag{3.54}$$

is fitted to the data. R(k) is the reliability of the program during the kth stage of testing. R(u) is the value of the R(k) as $k \to \infty$ and A is a growth parameter. If $A > 0$, the reliability of the program increases while $A < 0$, the reliability decreases.

To estimate the two unknowns R(u) and A, least squares estimates can be used. The desire is to minimize

$$S = \sum_{k=1}^{N} \left(R(k) - \frac{S_k}{n_k} \right)^2 \tag{3.55}$$

$$= \sum_{k=1}^{N} \left(R(u) - A/k - \frac{S_k}{n_k} \right)^2 . \tag{3.56}$$

The estimates which minimize this expression are found to be:

Estimates - Least Squares

$$\hat{A} = N \frac{\sum_{k=1}^{N} \frac{S_k}{n_k k} - \left(\sum_{k=1}^{N} \frac{S_k}{n_k} \right) \left(\sum_{k=1}^{N} \frac{1}{k} \right)}{\left(\sum_{k=1}^{N} \frac{1}{k} \right)^2 - N \left(\sum_{k=1}^{N} \frac{1}{k^2} \right)} \tag{3.57}$$

and

$$\hat{R}(u) = \frac{1}{N} \left[\hat{A} \sum_{k=1}^{N} \frac{1}{k} + \sum_{k=1}^{N} \frac{S_k}{n_k} \right] . \tag{3.58}$$

The only data then required to estimate the reliability curve are:

Data Requirement

The number of tests, n_i, performed at each stage and the number of successes observed at that stage, s_i.

The relationship of the reliability of the software to the stage number of the testing sequence is hard to justify. Moreover, a stage can have an arbitrary number of tests composing it. The only thing that marks the end of one stage and the beginning of another is some change to the program.

CHAPTER 4

TIME DOMAIN APPROACH

The Time Domain Approach to software reliability modeling has received the greatest emphasis in the applicable literature as it does in this report. This approach attempts to utilize either the times of error occurrences and the resulting times between error occurrences or the number of error occurrences per time period to model the error generation processes. In general, the models can be used to predict the expected time until the next error occurrence or the expected number of errors in the next interval of testing. These models were originally motivated by hardware reliability concepts and many of the terms used in hardware reliability modeling are carried over into the software.

Over the last 10 years, many models were proposed and extensions to them were given. There is still quite a lot of controversy about which is the "best" model to use on a software data set. Some studies were done comparing the various models on simulated and real data sets (see Chapter 5) and some studies are currently under way, but more research is needed. The best advise for applying these models to a software error data set is to apply a number of them to see which appears to best model the data; that is one purpose of this report. By providing a general overview of the various models, their assumptions, and data requirements; a number of models, which seem to be close to the actual way the data was generated, can be chosen. By applying some of these candidate models, the best model for a set of data can be established.

Section 4.1 discusses some of the hardware reliability concepts and terms that were adapted to software modeling. The difference in hardware versus software modeling is pointed out. Section 4.2 begins the discussion of software modeling with some of the classical adaptations of hardware concepts to software models. They are classical in the sense that many of these models are based on an exponential distribution for the time between error occurrence and the rate of error occurrence. The latter is determined by the number of errors in the program at the time of the test. Section 4.3 discusses the "Bayesian" philosophy applied to software modeling. This is followed by Section 4.4, which deals with attempting to model the behavior of the program as a Markov process.

The models chosen in this report were selected to provide the reader with an idea of the numerous approaches that have been proposed for software modeling. An extensive reference/bibliography is provided at the end of this report which may be of benefit to researchers in this area. An excellent report, giving an overview of software modeling in general and containing an extensive reference list, is a report written by Gephart, et. al.[18] This report is highly recommended for a researcher in this field.

4.1 HARDWARE VERSUS SOFTWARE RELIABILITY MODELING

In hardware reliability modeling,[19] a number of key concepts have been adapted for software modeling. The hazard rate Z(t) for a component (software program) is defined as the conditional probability that a failure (error) happens in an interval $(t,t+\Delta t)$ given that the component (program) has not failed up to time t. If T is the time when a failure (error) occurs, then:

$$Z(t)\Delta t = P\{t<T<t + \Delta t \mid T>t\} \quad . \tag{4.1}$$

The unconditional probability provides the failure (error) probability density function, f(t), for the component (software program); i.e.,

$$f(t)\Delta t = P\{t < T < t + \Delta t\} \quad . \tag{4.2}$$

The hazard function can be related to the pdf of the time of failure (error), f(t), as:

$$Z(t) = \frac{f(t)}{1 - F(t)} \tag{4.3}$$

where F(t) is the cumulative distribution of the time to failure; i.e.,

$$F(t) = \int_0^t f(x)dx. \tag{4.4}$$

The function

$$R(t) = 1 - F(t) \tag{4.5}$$

is called the reliability function of the component (program).

From equation (4.4), it can be seen that:

$$Z(x)dx = \frac{dF(x)}{1 - F(x)} \tag{4.6}$$

or

$$\int_0^t Z(x)dx = -\log\,[1 - F(x)]\Big|_0^t$$

$$- \int_0^t Z(x)dx = \log\,\frac{1 - F(t)}{1 - F(o)} \quad , \tag{4.7}$$

or

$$1 - F(t) = R(t) = \exp\,\{ - \int_0^t Z(x)dx\} \quad . \tag{4.8}$$

Thus once a hazard rate function for a component (program) is specified, the reliability function R(t) is then determined. Once the reliability function has been established, the expected time between failures (errors) or Mean Time Before Failure (MTBF) is calculated as:

$$\text{MTBF} = \int_0^\infty R(x)dx = \int_0^\infty tf(t)dt. \tag{4.9}$$

Many of the models in the next paragraph present forms for the hazard rate which, using the previous relationships, determine the reliability function for the program and the MTBF. Some of the models in Section 4.2 contain terms that do not have counterparts in hardware, e.g., the number of errors remaining and the time required to discover the remaining errors.

The concepts of hardware reliability modeling were adapted to software modeling. This is not to imply that the behavior of software is similar to hardware; quite the opposite is true. Software does not wear out over its life cycle as hardware does. In the reproduction of software, there is no generation of new random software errors introduced in subsequent copies. Duplicate software programs yield identical results. Moreover, software does not change during repeated operational use as hardware does. It is, in fact, that inconstant property of hardware upon which the probabilistic modeling of hardware failure occurrence is based. For software, it is the unchangeability over time that makes software error generation independent of time. The elapsing of a time variable does not cause software errors. For this reason, a number of researchers have strongly questioned the modeling of error occurrence in which time plays a factor. (See References 20 and 21.) It is not a direct relationship between a time variable and error generation that is modeled, however, but an indirect relationship as a result of the randomization of the input space for a program in operational use.

Within a program are latent errors which are discovered when a certain combination of input variables cause execution of the program to go down the path in which the error lies. Because of the very large number of combinations of input variables that are possible, the operational usage of a program gives the appearance of randomization over the input space. This, in turn, causes the error occurrences to take on the appearance of following a probabilistic model over time. It is characterization of this probabilistic nature on which the modeling is based.

4.2 CLASSICAL SOFTWARE MODELS

4.2.1 Weibull Model

Since a number of the concepts of hardware reliability theory were initially adapted to software, one of the earliest models to be applied was the Weibull Model. Because of the nature of the Weibull distribution, it can be used to

model increasing, decreasing, or constant failure rates for software. The form of the hazard rate is taken as:

$$Z(t) = \frac{a}{b}\left(\frac{t}{b}\right)^{a-1} \qquad \text{where a,b are constants} > 0 \text{ and } t \geq 0 \qquad (4.10)$$

so that if $a > 1$, the error rate increases with time; if $a < 1$, it decreases with time; and if $a = 1$, there is a constant failure rate over time. The corresponding pdf for the time to failure is the Weibull distribution; i.e.,

$$f(t) = \frac{a}{b}\left(\frac{t}{b}\right)^{a-1} \exp\left\{-\left(\frac{t}{b}\right)^{a}\right\} \qquad t \geq 0 \qquad (4.11)$$

with the cumulative distribution function

$$F(t) = \int_0^t f(x)dx = 1 - \exp\left\{-\left(\frac{t}{b}\right)^{a}\right\} \quad . \qquad (4.12)$$

(Notice that if $a = 1$, i.e., a constant failure rate, $f(t)$ becomes the exponential distribution.)

The reliability function is therefore:

$$R(t) = 1 - F(t) = \exp\left\{-\left(\frac{t}{b}\right)^{a}\right\} \quad , \qquad (4.13)$$

and hence, the MTBF is:

$$MTBF = \int_0^\infty R(t)dt = \int_0^\infty tf(t)dt = \frac{b}{a}\,\Gamma\left(\frac{1}{a}\right) \qquad (4.14)$$

where $\Gamma(\cdot)$ is the gamma function.

Coutinho[22] (also see Reference 18) proposes estimating the unknowns using as input the following data requirements.

Data Requirements

(a) The total number of errors in each time interval of testing

$$n_i,\ i = 1,\ldots,K\ ,$$

(b) The length of the testing interval

$$d_i,\ i = 1,\ldots,K\ ,$$

(c) The total number of time intervals, K, and

(d) The cumulative number of errors found to date

$$M = \sum_{i=1}^{K} n_i \quad .$$

The estimators are obtained using graphical procedures, method of moments, least squares, or MLEs. For instance, in the case of least squares, let

$$m = a, \tag{4.15}$$

$$b_o = -a\ell n(b), \tag{4.16}$$

$$F(i) = \frac{\sum_{j=1}^{i} n_j}{M} \quad \text{(normalized cumulative number of errors found up through the ith time interval)}, \tag{4.17}$$

and

$$X_i = \ell n\left(\sum_{j=1}^{i} d_j\right) . \tag{4.18}$$

From the expression of the cumulative distribution function

$$F(t) = 1 - \exp\left\{ - \left(\frac{t}{b}\right)^a \right\}, \tag{4.19}$$

we have

$$\frac{1}{1 - F(t)} = \exp\left\{ + \left(\frac{t}{b}\right)^a \right\}, \tag{4.20}$$

so

$$\ell n \left[\frac{1}{1 - F(t)} \right] = \left(\frac{t}{b}\right)^a \tag{4.21}$$

and thus,

$$\ell n \left[\ell n \left[\frac{1}{1 - F(t)} \right] \right] = a\ell n(t) - a\ell n(b); \tag{4.22}$$

that is,

$$y = mx + b_o \tag{4.23}$$

with

$$y = \ell n \left[\ell n \left[\frac{1}{1 - F(t)} \right] \right] \text{ and } x = \ell n(t). \tag{4.24}$$

The standard equation for a straight line, is thus obtained. If

$$Y_i = \ell n \left[\ell n \left[\frac{1}{1 - F(i)} \right] \right] , \tag{4.25}$$

there are n pairs of data points $(X_1,Y_1),\ldots,(X_n,Y_n)$ which (applying standard least squares) provide estimates of the slope and intercept; i.e.,

$$\hat{m} = \left[\frac{\sum_{i=1}^{K} (Y_i - \bar{Y})(X_i - \bar{X})}{\sum_{i=1}^{K} (X_i - \bar{X})^2} \right] , \tag{4.26}$$

and

$$\hat{b}_o = \bar{Y} - \hat{m}\,\bar{X} , \tag{4.27}$$

with

$$\bar{Y} = \frac{\sum_{i=1}^{K} Y_i}{K}$$

and (4.28)

$$\bar{X} = \frac{\sum_{i=1}^{K} X_i}{K} .$$

The estimates of a and b are then derived as:

Estimates

$$\hat{a} = \hat{m} \tag{4.29}$$

and

$$\hat{b} = \exp \left\{ - \frac{\hat{b}_o}{\hat{m}} \right\} . \tag{4.30}$$

The estimates of the reliability function and MTBF are given as:

$$\hat{R}(t) = \exp\left\{-\left(\frac{t}{\hat{b}}\right)^{\hat{a}}\right\} \tag{4.31}$$

and

$$\hat{MTBF} = \frac{\hat{b}}{\hat{a}}\ \Gamma\left(\frac{1}{\hat{a}}\right) . \tag{4.32}$$

Wagoner[23] (also see Reference 18) also applies the previous procedure to a set of software data, but suggests that the d_i's should be measured in CPU time rather than wall clock time. This a good suggestion and should be considered for all of the models. CPU time reflects the variation in testing effort from period to period. It also takes into account when no testing is going on. This is discussed again in relationship to Musa's Model.

4.2.2 Shooman Model

One of the earliest proposed software models was derived by Martin Shooman (References 24 through 29). The basic assumptions are:

Model Assumptions

(a) The number of errors in the code is a fixed number.

(b) No new errors are introduced into the code through the correction process.

(c) The number of machine instructions is essentially constant (i.e., the program is relatively mature).

(d) The detections of errors are independent.

(e) The software is operated in a similar manner as the anticipated operational usage.

(f) The error detection rate is proportional to the number of errors remaining in the code.

Suppose τ is the quantity of debugging time (in months) spent on the system since the start of the testing phase and suppose t is the operating time (measured in CPU) of the system. Using assumption (f) at any time t, the hazard rate is

$$Z(t) = K\varepsilon_r(\tau) \quad , \tag{4.33}$$

where K is the proportionality constant and $\varepsilon_r(\tau)$ is the error rate. This is taken as the number of errors remaining in the program, after τ months of debugging, normalized with respect to the total number of instructions in the code. This error rate, $\varepsilon_r(\tau)$, is mathematically expressed as:

$$\varepsilon_r(\tau) = \frac{E_T}{I_T} - \varepsilon_c(\tau) \tag{4.34}$$

where E_T is the total number of errors initially in the program; I_T is the number of machine instructions; and $\varepsilon_c(\tau)$ is the cumulative number of errors fixed in the interval from 0 to τ, normalized by the number of machine instructions. Since E_T and I_T are constant [assumptions (a) and (c)] and since no new errors are introduced in the correction process [assumption (b)] as:

$$\tau \to \infty \quad \varepsilon_c(\tau) \to \frac{E_T}{I_T} \quad , \tag{4.35}$$

so

$$\varepsilon_r(\tau) \to 0. \tag{4.36}$$

Combining equations (4.33) and (4.34),

$$Z(t) = K\left\{\frac{E_T}{I_T} - \varepsilon_c(\tau)\right\} . \tag{4.37}$$

Thus, the reliability function is:

$$R(t) = \exp\left\{- K\left[\frac{E_T}{I_T} - \varepsilon_c(\tau)\right] t\right\} \tag{4.38}$$

and the MTBF is:

$$MTBF = \frac{1}{K\left[\frac{E_T}{I_T} - \varepsilon_c(\tau)\right]} . \tag{4.39}$$

The only unknowns in this model are E_T and K. These quantities can be estimated in one of two ways.

The simplistic procedure is to use the moment technique. The required data inputs for this estimation procedure are given in the following.

Data Requirements - Moment Technique

Run a functional test of the program after two different debugging times, $\tau_1 < \tau_2$, which are chosen so that $\varepsilon_c(\tau_1) < \varepsilon_c(\tau_2)$, and record the following information:

(a) For each testing period, record the number of test runs that were made, i.e., r_1 and r_2 (usually $r_1 = r_2$).

(b) For each testing period and for each run, record the amount of CPU time that the program successfully executed. If out of the r_i runs made, m_i were successful with execution times $T_{i1},\ldots,T_{im}$, and $r_i - m_i$ were unsuccessful, but had successful execution times of $t_{i1},\ldots,t_{i,r_i - m_i}$ before the errors were discovered, then

$$H_i = \sum_{j=1}^{m_i} T_{ij} + \sum_{j=1}^{r_i - m_i} t_{ij} \tag{4.40}$$

is the total amount of successful execution time in the ith functional testing period.

The constant failure rate for the ith functional testing period is then estimated as:

$$\hat{Z}_i = \text{number of failures per hour} \tag{4.41}$$

$$= \frac{r_i - m_i}{H_i} . \tag{4.42}$$

Since the MTBF for a constant failure rate is the reciprocal of the failure rate, the MTBF for the ith functional testing period can be estimated as:

$$\hat{MTBF}_i = \frac{H_i}{r_i - m_i} . \tag{4.43}$$

If this expression is equated with the expression for MTBF, based on the model, it can be seen that:

$$\frac{H_1}{r_1 - m_1} = \hat{MTBF}_1 = \frac{1}{\left[K \frac{E_T}{I_T} - \varepsilon_c(\tau_1)\right]} , \tag{4.44}$$

and

$$\frac{H_2}{r_2 - m_2} = \hat{MTBF}_2 = \frac{1}{K\left[\frac{E_T}{I_T} - \varepsilon_c(\tau_2)\right]} \quad . \tag{4.45}$$

There are two equations in two unknowns, so solving for K and E_T, the estimates are obtained.

Estimates (Moment Estimators)

$$\hat{E}_T = I_T \frac{\left[(Z_2/Z_1)\ \varepsilon_c(\tau_1) - \varepsilon_c(\tau_2)\right]}{(Z_2/Z_1) - 1} \tag{4.46}$$

and

$$\hat{K} = \frac{Z_1}{\frac{E_T}{I_T} - \varepsilon_c(\tau_1)} \tag{4.47}$$

where

$$Z_i = \frac{r_i - m_i}{H_i} \quad . \tag{4.48}$$

The problem with this estimation procedure is the variation in the estimates as a function of the two debugging times, τ_1 and τ_2, chosen. Gephart et. al.[18] found that the estimates of E_T and K varied quite a bit depending on the two chosen points. They suggested that a number of pairs be chosen and the averages of the resulting estimates, using the previous equations, be used. Using the median of these derived estimators as a possible estimate could also be considered.

A second estimation procedure is based on the maximum likelihood procedure.[28,30] The data requirements are the same as required for the moments estimation procedure.

Data Requirements - Maximum Likelihood Estimation

(Same as moments techniques.)

The MLEs are:

Estimates (Maximum Likelihood Estimators)

$$\hat{E}_T = I_T \frac{\left[(Z_2/Z_1)\ \varepsilon_c(\tau_1) - \varepsilon_c(\tau_2)\right]}{(Z_2/Z_1) - 1} \tag{4.49}$$

and

$$\hat{K} = \left[\frac{1}{H_1 + H_2}\right] \left[\frac{r_1}{\frac{E_T}{I_T} - \varepsilon_c(\tau_1)} + \frac{r_2}{\frac{E_T}{I_T} - \varepsilon_c(\tau_2)}\right] \tag{4.50}$$

Notice that $\hat{E}_T$ is the same as the moments estimators. Here again, it is suggested that a number of pairs of points be chosen and the average or median of the resulting estimates of E_T and K be used.

4.2.3 Jelinski and Moranda "De-Eutrophication" Model

Another early model was one proposed by Jelinski and Moranda[31] while working for the McDonnell Douglas Astronautics Company. They developed this model for use on the Navy NTDS software and for a number of modules of the Apollo program. As can be seen in this paragraph, their work spawned quite a few variations of their basic model.

Model Assumptions

(a) The rate of error detection is proportional to the current error content of a program.

(b) All errors are equally likely to occur and are independent of each other.

(c) Each error is of the same order of severity as any other error.

(d) The error rate remains constant over the interval between error occurrences.

(e) The software is operated in a similar manner as the anticipated operational usage.

(f) The errors are corrected instantaneously without introduction of new errors into the program.

These assumptions are basically the same ones stated for Shooman's Model. (In fact, this report shows that the two models are equivalent when the correct correspondences are made.) The biggest questions are with regards to assumptions (c) and (f). It is difficult to envision a situation in which a perfect error correction process is achieved. The instantaneously corrected error part of the assumption can be avoided by not counting errors which were previously detected, but were not corrected. Assumption (c) can be avoided by dividing the errors into classes based upon severity. For instance, one might have a category for critical errors, a category for less serious errors, and one for minor errors (e.g., a misspelled word on an output). Software reliability models are then developed for each type. This approach is suggested.

Using assumptions (a), (b), (d), and (f), the hazard rate is defined as:

$$Z(t) = \phi\,[N - (i - 1)] \tag{4.51}$$

where t is any time point between the discovery of the (i - 1)th error and the ith error. The quantity ϕ is the proportionality constant given in assumption (a). N is the total number of errors initially in the system. Hence, if i - 1 errors have been discovered by time t, there are N - (i - 1) remaining errors so the hazard rate is proportional to this remaining number. Figure 4-1 is a plot of the hazard rate versus time. As can be seen, the rate is reduced by the same amount ϕ at the time of each error detection.

If $X_i = t_i - t_{i-1}$, i.e., the time between the discovery of the ith and the (i - 1)st error for i = 1,...,n where $t_0 = 0$, using assumption (d), the X_i's are assumed to have an exponential distribution with rate $Z(t_i)$. That is:

$$f(X_i) = \phi\,[N - (i - 1)]\,\exp\{-\phi[N - (i - 1)]X_i\} \tag{4.52}$$

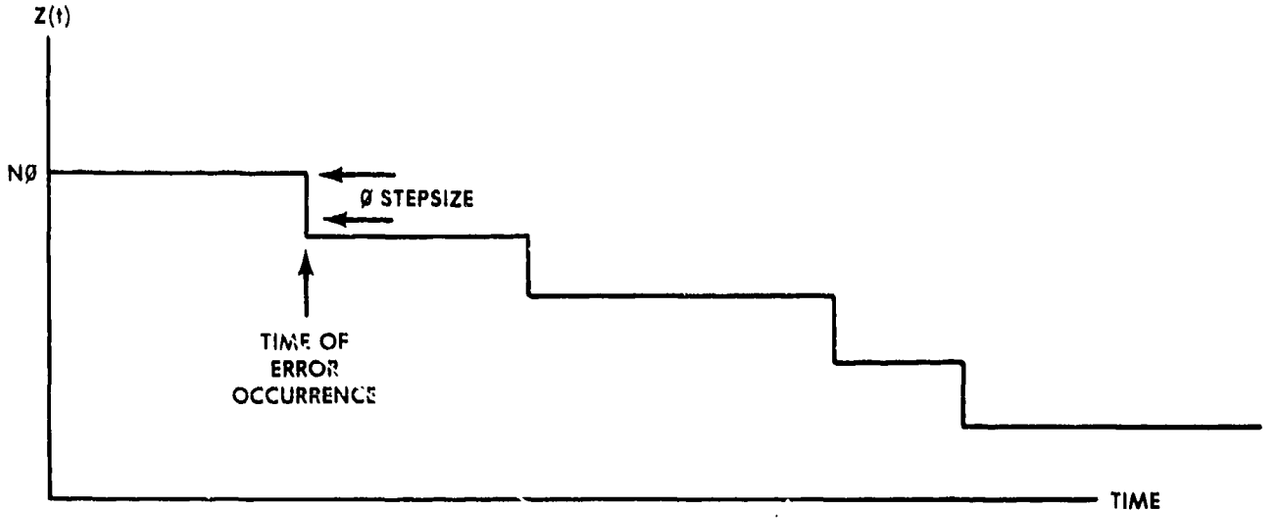

FIGURE 4-1. DE-EUTROPHICATION PROCESS

so the joint density for all the X_i's, using assumption (b), is:

$$L(X_1,\ldots,X_n) = \prod_{i=1}^{n} f(X_i) = \prod_{i=1}^{n} \phi\,[N - (i - 1)]\,\exp\{-\phi[N - (i - 1)]X_i\}\ . \qquad (4.53)$$

Taking the partial derivatives of ℓnL with respect to N and ϕ and setting the resulting equations equal to zero, the solutions for the following set of equations are obtained as MLEs for N and ϕ.

Estimates - Maximum Likelihood

$$\hat{\phi}_{ML} = \frac{n}{\hat{N}\left(\sum_{i=1}^{n} X_i\right) - \sum_{i=1}^{n} (i - 1)X_i} \quad \text{and} \qquad (4.54)$$

$$\sum_{i=1}^{n} \frac{1}{\hat{N}_{ML} - (i - 1)} = \frac{n}{\hat{N}_{ML} - \frac{1}{\sum_{i=1}^{n} X_i}\left(\sum_{i=1}^{n} (i - 1)X_i\right)}\ . \qquad (4.55)$$

Equation (4.55) is solved for $\hat{N}$ using numerical techniques (e.g., Newton-Raphson) and is then substituted into equation (4-54) to obtain an estimate of ϕ. The estimate of the MTBF is therefore derived after the jth error occurrence as:

$$\hat{MTBF}\ \{\text{for the (j+1)st error}\} = \frac{1}{\hat{Z}(t_j)} = \frac{1}{\hat{\phi}_{ML}(\hat{N}_{ML} - j)}\ . \qquad (4.56)$$

A report by Tal[32] derives the least squares estimators for N and ϕ as the estimators which minimize the sum of the squared differences between the observed time between failures and their mean values, i.e., the MTBFs. The quantity to be minimized is:

$$\sum_{i=1}^{n} (X_i - MTBF_i)^2 = \sum_{i=1}^{n} \left(X_i - \left(\frac{1}{\phi[N - (i - 1)]}\right)\right)^2\ . \qquad (4.57)$$

Again taking the partial derivatives of this expression with respect to ϕ and N and setting the resulting equations equal to zero, the least squares estimates are found to be the solutions to the following pair of equations:

Estimates - Least Squares

$$\hat{\phi}_{LS} = \frac{\sum_{i=1}^{n} \frac{1}{[\hat{N} - i + 1]^2}}{\sum_{i=1}^{n} \frac{X_i}{\hat{N} - i + 1}} \tag{4.58}$$

and

$$\left(\sum_{i=1}^{n} \frac{X_i}{(\hat{N}_{LS} - i + 1)^2}\right) \left(\sum_{i=1}^{n} \frac{1}{(\hat{N}_{LS} - i + 1)^2}\right) = \left(\sum_{j=1}^{n} \frac{X_i}{(\hat{N}_{LS} - i + 1)}\right) \left(\sum_{i=1}^{n} \frac{1}{(\hat{N}_{LS} - i + 1)^3}\right) \tag{4.59}$$

with the resulting estimate of the MTBF again being:

$$\hat{MTBF} \{\text{of the } (j + 1) \text{ error occurrence}\} = \frac{1}{\hat{Z}(t_i)} = \frac{1}{\hat{\phi}_{LS}[\hat{N}_{LS} - j]} . \tag{4.60}$$

Tal's[32] report also provides estimates for ϕ, N, and $\hat{MTBF}$ based upon a Least Squares Approach using the times of error occurrences, t_i's, rather than the time between error occurrences, X_i's. It states that the t's are integrals of the X's, $t_i = \sum_{j=1}^{i} X_j$, and hence the estimates tend to behave better as the t_i's fluctuate less due to the cumulative summary effect.

The estimates for ϕ and N are derived by minimizing the sum of squared deviations of:

$$\sum_{i=1}^{n} (t_i - \text{expected})^2 = \sum_{i=1}^{n} \left(t_i - \sum_{j=1}^{i} \frac{1}{\phi(N - j + 1)}\right)^2 . \tag{4.61}$$

Taking partials and setting the resulting equations equal to zero, the estimates, based upon the times of error occurrences, are found to be the solutions of the following equations:

$$\hat{\phi}_t = \frac{\sum_{i=1}^{n} A_i^2}{\sum_{i=1}^{n} t_i A_i} \tag{4.62}$$

with

$$A_i = \sum_{j=1}^{i} \frac{1}{\hat{N}_t - j + 1} \tag{4.63}$$

and

$$\sum_{i=1}^{n} t_i B_i \left(\sum_{i=1}^{n} A_i^2\right) = \left(\sum_{i=1}^{n} t_i A_i\right)\left(\sum_{i=1}^{n} A_i B_i\right) \tag{4.64}$$

with

$$B_i = \sum_{j=1}^{i} \frac{1}{(\hat{N}_t - j + 1)^2} . \tag{4.65}$$

Again the estimated MTBF, after observing j failures, is:

$$\hat{\text{MTBF}}\ \{(j+1)\text{ error occurrence}\} = \frac{1}{\hat{\phi}_t(\hat{N}_t - j)} . \tag{4.66}$$

Using the various estimates of $\hat{\text{MTBF}}$, the estimated time to remove the next m errors, after observing n failures, can be derived. Using any of the previous estimates for ϕ and N, the estimate is obtained as:

Estimated Time to Remove the Next m Errors

$$= \sum_{j=n+1}^{n+m} \hat{\text{MTBF}}\ \{j\text{ error occurrence}\} \tag{4.67}$$

$$= \sum_{j=n+1}^{n+m} \frac{1}{\hat{\phi}\hat{N} - j + 1)} . \tag{4.68}$$

The only data required for the calculation of the estimates are:

Data Requirements

(a) Either the time between error occurrence (x_i's) or

(b) The time of error occurrence (t_i's).

Once one is recorded, the other ($x_i = t_i - t_{i-1}$) is obtained.

A number of authors have derived the large sample approximations to the variances of these estimates (see References 32 and 33, also 17*) using the asymptotic properties of the MLEs. It can be shown for large n and N that:

$$\text{var}\left\{\hat{\phi}_{ML}\right\} = \frac{\sum_{i=1}^{n} \frac{1}{(N - i + 1)^2}}{D} \tag{4.69}$$

$$\text{var}\left\{\hat{N}_{ML}\right\} = \frac{n}{\phi^2 D} \tag{4.70}$$

$$\text{cov}\left(\hat{N}_{ML}, \hat{\phi}_{ML}\right) = \frac{-\sum_{i=1}^{n} x_i}{D} \tag{4.71}$$

where

$$D = \frac{\sum_{i=1}^{n} \frac{n}{(N - i + 1)^2}}{\phi^2} - \left(\sum_{i=1}^{n} x_i\right)^2 \tag{4.72}$$

and

$$\text{var}\left\{\hat{MTBF}_{ML} \text{ after the nth error occurrence}\right\} =$$

$$\frac{1}{C}\left\{\sum_{i=1}^{n} \frac{1}{(N - i + 1)^2} - \frac{n}{(N - n)^2} + \frac{2E\phi}{(N - n)^2}\right\} \tag{4.73}$$

where

$$C = \left\{\sum_{i=1}^{n} \frac{1}{(N - i + 1)^2} - \frac{n}{(N - n)^2} - \frac{2E\phi}{(N - n)^2}\right\}\left\{-n\phi^2(N - n)^2 + \right.$$

$$\left. + 2\phi^3(N - n)^3\left(\sum_{i=1}^{n} x_i\right) + 2E\phi^3(N - n)^2\right\} - E^2\phi^4 \tag{4.74}$$

and

$$E = \sum_{i=1}^{n} (n - i + 1)x_i \ . \tag{4.75}$$

---

* There is an error for the variance of N in this paper.

Using these various estimates and variances, approximate $(1 - \alpha) \times 100$ percent confidence intervals can be constructed for the corresponding population parameters as:

Confidence Intervals

$$\left(\hat{\phi}_{ML} - Z_{1-\frac{\alpha}{2}} \sqrt{\text{var}\{\hat{\phi}_{ML}\}}, \quad \hat{\phi}_{ML} + Z_{1-\frac{\alpha}{2}} \sqrt{\text{var}\{\hat{\phi}_{ML}\}} \right) \tag{4.76}$$

$$\left(\hat{N}_{ML} - Z_{1-\frac{\alpha}{2}} \sqrt{\text{var}\{\hat{N}_{ML}\}}, \quad \hat{N}_{ML} + Z_{1-\frac{\alpha}{2}} \sqrt{\text{var}\{\hat{N}_{ML}\}} \right) \tag{4.77}$$

$$\left(\hat{MTBF}_{ML} - Z_{1-\frac{\alpha}{2}} \sqrt{\text{var}\{\hat{MTBF}_{ML}\}}, \quad \hat{MTBF}_{ML} + Z_{1-\frac{\alpha}{2}} \sqrt{\text{var}\{MTBF_{ML}\}} \right) . \tag{4.78}$$

Any unknowns in the expressions for the variances are replaced by their MLEs and $Z_{1-\frac{\alpha}{2}}$ is the point taken from a standard normal table such that $P\{Z \geq Z_{1-\frac{\alpha}{2}}\} = \frac{\alpha}{2}$. Large sample confidence intervals can be constructed using the least squares estimates. Schafer, et.al. (see Reference 33) use the result that if $X_1, \ldots, X_n$ are independent random variables with finite moments:

$$E\{X_i\} = g_i(\theta_1, \theta_2), \ \text{var}\{X_i\} = \sigma_i^2(\theta_1, \theta_2); \tag{4.79}$$

$$E\{(X_i - g_i(\theta_1, \theta_2))^3\} = \rho_i(\theta_1, \theta_2) , \tag{4.80}$$

which satisfies Liapunov's condition, that is,

$$\lim_{n \to \infty} \frac{\left\{ \sum_{i=1}^{n} \rho_i(\theta_1, \theta_2) \right\}^{1/3}}{\left\{ \sum_{i=1}^{n} \sigma_i^2(\theta_1, \theta_2) \right\}^{1/2}} = 0 \tag{4.81}$$

for each finite value of the unknown parameters θ_1 and θ_2, g_i is three times continuously differentiable in a neighborhood of the true values for θ_1 and θ_2, then the least squares estimators of θ_1 and θ_2, obtained by minimizing:

$$S(\theta_1, \theta_2) = \sum_{i=1}^{n} [X_i - g_i(\theta_1, \theta_2)]^2 , \tag{4.82}$$

are (subject to certain restrictions on the partial derivatives of g_i) asymptotically bivariate normal with mean vector (θ_1,θ_2) and covariance matrix:

$$\frac{1}{\Delta^2}\sum_1\begin{pmatrix}\sum_{i=1}^{n}\sigma_i^2(\theta_1,\theta_2)\left(\frac{\partial g_i}{\partial\theta_1}\right)^2 & \sum_{i=1}^{n}\sigma_i^2(\theta_1,\theta_2)\frac{\partial g_i}{\partial\theta_1}\frac{\partial g_i}{\partial\theta_2}\\ \sum_{i=1}^{n}\sigma_i^2(\theta_1,\theta_2)\frac{\partial g_i}{\partial\theta_1}\frac{\partial g_i}{\partial\theta_2} & \sum_{i=1}^{n}\sigma_i^2(\theta_1,\theta_2)\left(\frac{\partial g_i}{\partial\theta_2}\right)^2\end{pmatrix}\sum_1 \tag{4.83}$$

where

$$\Delta = \sum_{i=1}^{n}\left(\frac{\partial g_i}{\partial\theta_1}\right)^2\sum_{i=1}^{n}\left(\frac{\partial g_i}{\partial\theta_2}\right)^2 - \left(\sum_{i=1}^{n}\frac{\partial g_i}{\partial\theta_1}\frac{\partial g_i}{\partial\theta_2}\right)^2 \tag{4.84}$$

and

$$\sum_1 = \begin{pmatrix}\sum_{i=1}^{n}\left(\frac{\partial g_i}{\partial\theta_2}\right)^2 & -\sum_{i=1}^{n}\frac{\partial g_i}{\partial\theta_1}\frac{\partial g_i}{\partial\theta_2}\\ -\sum_{i=1}^{n}\frac{\partial g_i}{\partial\theta_1}\frac{\partial g_i}{\partial\theta_2} & \sum_{i=1}^{n}\left(\frac{\partial g_i}{\partial\theta_1}\right)^2\end{pmatrix}. \tag{4.85}$$

(See Reference 33 for details.)

If the X_i's are the times between error occurrences, then

$$E\{X_i\} = \frac{1}{\phi[N-(i-1)]} = g_i(N,\phi) \tag{4.86}$$

and

$$\mathrm{Var}\{X_i\} = \sigma_i^2(N,\phi) = \frac{1}{\phi^2[N-(i-1)]^2} \tag{4.87}$$

with $\theta_1 = N$ and $\theta_2 = \phi$ in the previous statement of the theorem.

Thus,

$$\frac{\partial g_i}{\partial N} = \frac{\partial g_i}{\partial\theta_1} = \frac{-1}{\phi[N-(i-1)]^2} \tag{4.88}$$

and

$$\frac{\partial g_i}{\partial \phi} = \frac{\partial g_i}{\partial \theta_2} = \frac{-1}{\phi^2[N - (i - 1)]} \quad . \tag{4.89}$$

If the previous expressions are substituted into the asymptotic covariance matrix, $(1 - \alpha)$ X 100 percent confidence intervals can be formulated for ϕ and N, replacing any unknowns by their least squares estimates. A similar approach can be taken to derive confidence intervals for ϕ and N based upon the actual observed times of failure, t_i's, but it is not developed here.

In seeking the estimates and their resulting confidence intervals, the biggest problem faced is the difficulty in convergence of the numerical techniques employed to find the MLEs or least squares estimates. Difficulties encountered include (see Reference 33) lack of convergence, sensitivity of the iteration scheme to the starting value, convergence to a saddlepoint or invalid estimate, and nonuniqueness of the estimates. The choice of a starting point was especially critical to the maximum likelihood procedures. Littlewood and Verrall[34] have shown that a unique maximum at finite N and nonzero ϕ is attained if and only if

$$\frac{\sum_{i=1}^{n} (i - 1)X_i}{\sum_{i=1}^{n} (i - 1)} > \frac{\sum_{i=1}^{n} X_i}{n} \quad ; \tag{4.90}$$

otherwise, the MLE for N is ∞. Essentially this condition means that the model can only be applied to software that exhibits software growth, i.e., $X_i > X_{i-1}$. In any computer implementation of this model, the previous condition should first be verified to ensure a unique finite maximum exists. Another problem with the MLE of N was pointed out by Forman and Singpurwalla[35] concerning the instability of the estimate. If

$$\frac{\sum_{i=1}^{n} (i - 1)X_i}{\sum_{i=1}^{n} X_i} \tag{4.91}$$

is small, there is the problem pointed out by Littlewood and Verrall,[34] but if it is large (so that the times between failures during the latter stages of testing

are greater than the ones during the earlier stages), a new problem with the estimate arises. The MLE, $\hat{N}_{ML}$, tends to be close to n, the number of errors found to date. This tends to give a more optimistic view of current reliability of the program. It gives the impression that the program is very close to being errorless when in fact the real error count N may be much larger than n.

To overcome this drawback, Forman and Singpurwalla suggest that the behavior of the likelihood function be examined in greater detail. In particular, they suggest the following procedure be employed as a stopping rule:

(a) Calculate the MLE of N.

(b) If $\hat{N}_{ML} \tilde{\sim} n$, go to step (c); if $\hat{N}_{ML} >> n$, observe another failure interval X_{n+1} and go back to step (a).

(c) Compute

$$R(N) = \frac{L(N, \hat{\phi}(N))}{L(N_{ML}, \phi_{ML})} \tag{4.92}$$

where L is the likelihood function

$$L(N,\phi) = \prod_{i=1}^{n} \phi[N - i + 1] \exp(-\phi[N - i + 1]x_i) \tag{4.93}$$

and

$$\hat{\phi}(N) = \frac{n}{(N \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} (i-1)x_i)} \tag{4.94}$$

(d) Compute

$$R_{NORMAL}(N) = \exp[-\tfrac{1}{2}(\hat{N}_{ML} - N)^2 / \text{var}(\hat{N}_{ML})] \tag{4.95}$$

using the formula for $\text{var}(\hat{N}_{ML})$ given earlier.

(e) Compare R(N) and $R_{NORMAL}(N)$ for various values of N. If they agree well, then $\hat{N}_{ML} \tilde{\sim} n$ is a good estimator of N. If they do not agree, then $\hat{N}_{ML}$ is a misleading estimator of N; observe another failure time interval and go to step (a).

These suggestions should also be employed in any analysis utilizing this model.

In Forman and Singpurwalla's paper, a procedure is described for testing the hypothesis that no errors remain in the program and an optimal time interval for testing is developed based upon cost. The test of the null hypothesis

$$H_o: \quad N = n, \text{ i.e., no errors remain in the program}$$

versus

$$H_a: \quad N > n$$

is performed by exercising the software under operational conditions for an additional t_{length} of time. If a failure is observed, the null hypothesis is rejected; otherwise, it is accepted. The additional length of time is estimated as:

$$t_{length} = \frac{-\ell n(1 - \beta)}{\hat{\phi}_{ML}} \tag{4.96}$$

where β is the desired power of the test, i.e., the probability of rejecting the null hypothesis when it is false. The actually achieved power is at least β if $\hat{\phi}_{ML}$ is close to ϕ.

The optimal additional time, t_{length}, of testing based upon cost and mission time, t_m, is constructed as follows. If the software fails during the additional testing time, the cost incurred is

$$C_1(t) \quad \text{where} \quad 0 < t < t_{length} \tag{4.97}$$

and $C_1(t)$ is a convex nondecreasing function of time representing the cost incurred in testing for time t. If the software passes the additional testing time, but fails in operational use, the cost is

$$C_1(t_{length}) + C_2 \quad \text{for} \quad t_{length} \leq t \leq t_{length} + t_m \text{ , where} \tag{4.98}$$

C_2 is a fixed cost due to an operational failure of the software.

If no software error is encountered during either testing or mission time, the total cost is

$$C_1(t_{length}) \quad \text{for} \quad t_{length} + t_m < t. \tag{4.99}$$

The total expected cost is therefore:

$$E\{C\} = \int_0^{t_{length}} C_1(t)\phi e^{-\phi t}dt + \int_{t_{length}}^{t_{length}+t_m} (C_1(t_{length}) + C_2)\,\phi e^{-\phi t}dt + \int_{t_{length}+t_m}^{\infty} C_1(t_{length})\phi e^{-\phi t}dt \quad . \tag{4.100}$$

It can be shown that when:

$$\left.\frac{dC_1(t)}{dt}\right|_{t=o} \geq \phi\, C_2 \exp\{\phi t_m\} \quad , \tag{4.101}$$

$t_{length} = 0$ minimizes the expected cost. This means that if the additional cost of testing is more than the cost of an operational failure, no additional testing should be done. If, however,

$$\left.\frac{dC_1(t)}{dt}\right|_{t=o} < \phi\, C_2 \exp\{\phi t_m\} \quad , \tag{4.102}$$

then

$$t_{length} = \left(\left.\frac{dC_1(t)}{dt}\right|_{t = \phi\, C_2 \exp(\phi t_m)} \right)^{-1} \tag{4.103}$$

minimizes $E\{C\}$.

A final point concerning the Jelinski-Moranda Model[31] is that it is equivalent to Shooman's Model. In Paragraph 4.2.2, Shooman's Model was given as:

$$Z(t) = K\left\{\frac{E_T}{I_T} - \varepsilon_c(\tau)\right\} \tag{4.104}$$

where K is a proportionality constant, I_T is the total number of instructions, E_T is the total number of errors initially in the program, and $\varepsilon_c(\tau)$ is the cumulative number of errors corrected in the interval 0 to τ, normalized by the number of machine instructions. Noting that $E_T = N$ in the Jelinski-Moranda Model and

$$\varepsilon_c(\tau) = \frac{i-1}{I_T} \text{ for all } t_{i-1} \leq \tau \leq t_i \quad , \tag{4.105}$$

the hazard rate function for the Shooman Model is derived. It is

$$Z(t) = K\left\{\frac{N}{I_T} - \frac{i-1}{I_T}\right\} \quad \text{for} \quad t_{i-1} \leq t \leq t_i \tag{4.106}$$

$$= \frac{K}{I_T} \{N - (i-1)\} \tag{4.107}$$

$$= \phi \{N - (i-1)\} \quad , \tag{4.108}$$

letting $\phi = \frac{K}{I_T}$. This is precisely the hazard rate for the Jelinski-Moranda Model just considered.

The next few paragraphs of this report present, in various amounts of detail, extensions that have been made to the basic model.

4.2.3.1 Jelinski and Moranda's Model 1 and Model 2. Jelinski and Moranda's basic model cannot be applied to software programs which are not complete. The program must be relatively stable with a total of N errors present initially in the code. Their first extension of the basic model[36] is for programs that are undergoing development. If at any point in time an error is discovered, an estimate of the reliability based upon the percentage completed for the module or program can be given. Specifically, they let S(t) be the nondecreasing fraction of the total number of statements which a complete program has, measured at time t, where t is either elapsed wall clock or CPU time. Thus $S(0) = 0$ and $S(T_{END}) = 1$, where T_{END} is the end time of the program development. The only requirement about the nature of the function S(t) is that it be nondecreasing, its values be known at the times of error occurrence $(t_1, t_2, \ldots, t_n)$, and that it be constant during the times between error occurrence. The hazard rate is then formulated as

$$Z_i(t) = \phi S_{i-1} [N - i + 1] \quad \text{for} \quad t_{i-1} \leq t \leq t_i \quad i = 1, \ldots, n \quad . \tag{4.109}$$

Here N is interpreted as the error content at the end of program development, i.e., when $S(T_{END}) = 1$. S_{i-1} is the fraction of the program which was completed prior to the start of the ith interval, i.e., $S_{i-1} = S(t_{i-1})$. The likelihood function is then:

$$L(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} \phi S_{i-1}[N - i + 1] \exp\{-\phi S_{i-1}[N - i + 1]X_i\} \tag{4.110}$$

where again

$$X_i = t_i - t_{i-1} \quad . \tag{4.111}$$

The MLEs are obtained as the solutions to the following system of equations.

Model 1
Estimates - Maximum Likelihood

$$\hat{\phi}_{ML,1} = \frac{\sum_{i=1}^{n} \frac{1}{\hat{N}_{ML,1} - i + 1}}{\sum_{i=1}^{n} S_{i-1}X_i} \tag{4.112}$$

and

$$\sum_{i=1}^{n} S_{i-1}(\hat{N}_{ML,1} - i + 1)X_i = \frac{\sum_{i=1}^{n} S_{i-1}X_i}{\sum_{i=1}^{n}\left(\frac{1}{\hat{N}_{ML,1} - i + 1}\right)}. \tag{4.113}$$

Notice that if $S_{i-1} = 1$ for all i, then the MLEs ar the same as in the previous paragraph. The MLEs are very similar to the ones obtained for the basic model in Paragraph 4.2.3. In these equations there is $S_{i-1}X_i$, while in the previous section there were X_i's. The S_{i-1} has the effect of reducing the time between the ith and (i - 1)st error occurrence by the same fraction as the percentage of the program completed.

The MLE of the MTBF, after n errors have been observed, is easily established as:

$$\hat{MTBF}_1 = \frac{1}{S(t_n)(\hat{N}_{ML,1} - n)\hat{\phi}_{ML,1}}. \tag{4.114}$$

Model 2^{36} also allows for a developing program, but the requirement of knowing the fraction completed at each stage is eliminated. For a developing program, it is hard to envision a case where the manager knows with certainty the size of the end program. Moreover the assumption,

$$S(t) = S_{i-1} \qquad t_{i-1} \leq t < t_i , \tag{4.115}$$

i.e., a constant function between times of error occurrence, is unrealistic. The very nature of a developing program dictates a continuously changing function of time.

For Model 2, the assumption is made that the error-making rate for the programming team (E_p) is constant over the time of program development. The hazard rate at time t is then taken to be:

$$Z(t) = \phi[G_{i-1} E_p - (i - 1)], \quad \text{for} \quad t_{i-1} \leq t \leq t_i \tag{4.116}$$

where ϕ is the constant of proportionality, and G_{i-1} is the number of lines of code developed by the time of the (i - 1)st error occurrence. Again the basic assumption is that the rate of error occurrence is proportional to the number of errors remaining in the code. $G_{i-1} E_p$ is the total error count present in the G_{i-1} lines of code; of which i - 1 have been found. The likelihood function is again expressed as (using the model assumptions of the previous section):

$$L(X_1,\ldots,X_n) = \prod_{i=1}^{n} \phi\,[G_{i-1} E_p - (i - 1)] \exp\,(-\phi X_i[G_{i-1} E_p - (i-1)]). \tag{4.117}$$

The MLEs of ϕ and E_p are obtained as the solution to the resulting systems of equations.

Model 2
Estimates - Maximum Likelihood

$$\hat{\phi}_{ML,2} = \frac{\displaystyle\sum_{i=1}^{n} \frac{G_{i-1}}{G_{i-1}\hat{E}_p - (i - 1)}}{\displaystyle\sum_{i=1}^{n} G_{i-1}X_i} \tag{4.118}$$

and

$$\sum_{i=1}^{n} [G_{i-1}\hat{E}_p - (i - 1)]X_i = \frac{n \displaystyle\sum_{i=1}^{n} G_{i-1} X_i}{\displaystyle\sum_{i=1}^{n} \frac{G_{i-1}}{G_{i-1}\hat{E}_p - (i - 1)}}. \tag{4.119}$$

An estimate of the MTBF is then obtained as:

$$\hat{MTBF}_2 = \frac{1}{\hat{\phi}_{ML,2}\,[G_n\hat{E}_p - n]}. \tag{4.120}$$

In the early stages of development of the program, the assumption that E_p is constant is questionable. As the programmers experience a learning curve phenomena the error rate is expected to go down. Moreover, if the programmers' team experiences a turnover in personnel, with inexperienced people being hired at various points in the program development, it seems hard to justify a constant E_p. The model seems suitable if the development time frame can be broken up into smaller time regions over which E_p can be taken to be constant.

4.2.3.2 Lipow's Extension Model. Lipow[37] proposed an extension to the Jelinski-Moranda Model by allowing more than one error occurrence during an interval of testing and also allowing that all errors found in a given testing interval need not be corrected by the start of the next testing period. Specifically, the model assumptions are:

Model Assumptions

(a) The rate of error detection is proportional to the current error content of a program.

(b) All errors are equally likely to occur and are independent of each other.

(c) Each error is of the same order of severity as any other error.

(d) The error rate remains constant over the testing interval.

(e) The software operates in a similar manner as the anticipated operational usage.

(f) During a testing interval i, f_i errors are discovered but only n_i are corrected in the time frame.

The previous assumptions are identical to the assumptions of the Jelinski-Moranda Model except for (f). Suppose there are M periods of testing in which testing interval i is of length x_i. During this time frame, f_i errors are discovered, of which n_i are corrected. Assuming the error rate remains constant during each of the M testing periods [assumption (d)], the hazard rate during the ith testing period is:

$$Z(t) = \phi[N - F_{i-1}] \quad t_{i-1} \leq t \leq t_i \quad ; \tag{4.121}$$

where

ϕ is the proportionality constant,

N is again the total number of errors initially present in the program.

$F_{i-1} = \sum_{j=1}^{i-1} n_j$ is the total number of errors corrected up through the (i-1)st testing intervals, and t_i is the time measured in either CPU or wall clock time of the end of the ith testing interval ($x_i = t_i - t_{i-1}$). The t_i's are fixed and thus, are not random as in the Jelinski-Moranda Model. Taking the number of failures, f_i, in the ith interval to be a Poisson random variable with mean $Z(t_i)x_i$, the likelihood is:

$$L(f_1,\ldots,f_M) = \prod_{i=1}^{M} \frac{[\phi\,[N - F_{i-1}]x_i\,]^{f_i} \exp\{-\phi[N - F_{i-1}]x_i\}}{f_i!} \quad . \quad (4.122)$$

Taking the partial derivatives with respect to ϕ and N of ℓn L and setting the resulting equations equal to zero, the MLEs can be obtained as solutions to the following system of equations:

Estimates - Maximum Likelihood

$$\hat{\phi}_L = \frac{F_M/A}{N_L + 1 - B/A} \quad (4.123)$$

and

$$\frac{F_M}{N_L + 1 - B/A} = \sum_{i=1}^{M} \frac{f_i}{N_L - F_{i-1}} \quad (4.124)$$

where

$F_M = \sum_{i=1}^{M} f_i$, the total number of errors found in the M periods of testing,

$$B = \sum_{i=1}^{M} (F_{i-1} + 1)x_i \quad , \quad (4.125)$$

and

$$A = \sum_{i=1}^{M} x_i \quad , \quad (4.126)$$

the total length of the testing periods. From these MLEs, the maximum likelihood of the mean time until the next failure is:

$$\hat{MTBF}_L = \frac{1}{\hat{\phi}_L(N_L - F_M)} \quad . \quad (4.127)$$

Lipow gives the asymptotic variance of the estimates as:

$$\text{var}\{\hat{\phi}_L\} = \left(\sum_{i=1}^{M} \frac{f_i}{(N_L - F_{i-1})^2} \right) / D \quad , \tag{4.128}$$

$$\text{var}\{\hat{N}\} = \frac{F_M}{\phi_L^2 D} \quad , \tag{4.129}$$

and

$$\text{cov}(\hat{N}_L, \hat{\phi}_L) = - \sum_{i=1}^{M} x_i / D \tag{4.130}$$

where

$$D = \frac{F_M}{\phi_L^2} \left(\sum_{i=1}^{M} \frac{f_i}{(N_L - F_{i-1})^2} \right) - \left(\sum_{i=1}^{M} x_i \right)^2 \quad . \tag{4.131}$$

Notice that if $f_i = 1$ $i = 1,\ldots,M$ i.e., only one error is discovered per time period, and $F_{i-1} = i - 1$ so that all errors are corrected upon discovery, then all of the estimates and their variances are precisely the formulas derived for the Jelinski-Moranda Model.

Using the previous formulas, large sample confidence intervals can be given for ϕ and N as:

100 x (1 - α) percent:
Confidence Intervals

$$\left(\hat{\phi}_L - Z_{1-\frac{\alpha}{2}} \sqrt{\text{var}\{\hat{\phi}_L\}} \; , \; \hat{\phi}_L + Z_{1-\frac{\alpha}{2}} \sqrt{\text{var}\{\hat{\phi}_L\}} \right) \tag{4.132}$$

and

$$\left(\hat{N}_L - Z_{1-\frac{\alpha}{2}} \sqrt{\text{var}\{\hat{N}_L\}} \; , \; \hat{N}_L + Z_{1-\frac{\alpha}{2}} \sqrt{\text{var}\{\hat{N}_L\}} \right) \tag{4.133}$$

with $Z_{1-\frac{\alpha}{2}}$ chosen from a standard normal table so that:

$$P\{Z \geq Z_{1-\frac{\alpha}{2}}\} = \frac{\alpha}{2} \; . \tag{4.134}$$

Note that if

$$f_j = n_j \quad j=1,\ldots, M \quad ; \tag{4.135}$$

i.e., the number of errors corrected in the ith interval is the same as the number discovered, then the previous model reduces to a model considered by Sukert[17], Gephart et.al.[18] and Lipow.[38]

4.2.3.3 Rushforth, Staffanson, and Crawford's Model. The last model considered as an extension of the Jelinski-Moranda Model is a model by Rushforth, Staffanson, and Crawford.[39] This model was originally given as an extension to an error generation model proposed by Shooman.[28] A model proposed by Tal and Barber[40] is also very similar to the one discussed. The basic idea for this class of models is to relax the assumption that the error correction process is perfect. This class allows for the introduction of new errors into the program in the correction of inherent ones. The specific assumptions for these models are given in the following.

Model Assumptions

(a) The rate of error detection is proportional to the current error content of the program.

(b) All errors are equally likely to occur and are independent of each other.

(c) Each error is of the same order of severity as any other error.

(d) The error rate remains constant over the testing interval of the particular program version undergoing testing.

(e) The software is operated in a similar manner as the anticipated operational usage.

(f) No attempt is made to correct detected errors at the time of error occurrence. Instead, at specified points in time $t_1, t_2,\ldots,t_j,\ldots$ a new corrected version of the program is provided.

(g) Of the detected errors reported, some are corrected, some are not, and some in the correction process cause the introduction of new errors.

(h) The error correction rate, $r_c(t)$, is proportional to both the error detection rate, $r_d(t)$, and the error backlog $n_b(t)$, defined as the difference between the number of errors detected by time t minus the number of errors corrected at that time. Specifically, $r_c(t)$ is taken as:

$$r_c(t) = \alpha\, r_d(t) + \beta n_b(t). \tag{4.136}$$

(i) The rate of error generation is assumed proportional to the error-correction rate.

$$r_\varepsilon(t) = \gamma\, r_c(t). \tag{4.137}$$

(j) The error detection process, $n_d(t)$, is completely known.

From these assumptions, Rushforth, Staffanson, and Crawford's Model is formulated as shown in the following.

For any model version j,

$$N_j = N - n_c(t_{j-1}) + n_\varepsilon(t_{j-1}) \quad , \tag{4.138}$$

where

N_j is the total number of errors present in version j,

N is the initial number of errors present,

$n_c(t_{j-1})$ is the total number of errors corrected up through program version j-1, and

$n_\varepsilon(t_{j-1})$ is the total number of errors introduced into the program in the correction of the previous (j - 1) versions. From assumption (a),

$$r_d(t) = \phi\, N_j \qquad \text{for all t's } t_{i-1} \le t \le t_i \quad , \tag{4.139}$$

where ϕ is the proportionality constant. From assumption (h), if $n_c(t)$ is the number of errors detected up through time t, then,

$$n_b(t) = n_d(t) - n_c(t) \tag{4.140}$$

so that:

$$r_c(t) = \alpha\, r_d(t) + \beta\, (n_d(t) - n_c(t)). \tag{4.141}$$

Finally, using the fact that:

$$n_\varepsilon(t) = \int_o^t r_\varepsilon(x)dx \tag{4.142}$$

and

$$n_c(t) = \int_o^t r_c(x)dx \quad , \tag{4.143}$$

it can be drawn from assumption (i) that:

$$n_\varepsilon(t) = \int_o^t r_\varepsilon(x)dx = \gamma \int_o^t r_c(x)dx = \gamma\, n_c(t). \tag{4.144}$$

Hence,

$$r_d(t) = \phi N_j \tag{4.145}$$

$$= \phi\, [N - n_c(t_{j-1}) + n_\varepsilon\, (t_{j-1})] \tag{4.146}$$

$$= \phi\, [N - n_c(t_{j-1}) + \gamma\, n_c\, (t_{j-1})] \tag{4.147}$$

$$= \phi\, [N - (1 - \gamma)\, n_c\, (t_{j-1})] \qquad t_{j-1} \le t \le t_j \tag{4.148}$$

and

$$r_c(t) = \alpha\, r_d\, (t_{j-1}) + \beta[n_d(t_{j-1}) - n_c(t_{j-1})] \tag{4.149}$$

$$= \alpha\, \phi\, N_j + \beta\, [n_d\, (t_{j-1}) - n_c\, (t_{j-1})] \tag{4.150}$$

$$= \alpha\, \phi\, [N - (1-\gamma)\, n_c\, (t_{j-1})] + \beta\, [n_d\, (t_{j-1}) - n_c\, (t_{j-1})]$$

$$t_{j-1} \le t \le t_j \ . \tag{4.151}$$

If

$$\phi_a = (1 - \gamma)\phi \tag{4.152}$$

and

$$N_a = N/(1 - \gamma)\ , \tag{4.153}$$

the two equations become:

$$r_d(t) = \phi_a\, [N_a - n_c\, (t_{j-1})] \tag{4.154}$$

and

$$r_c(t) = \alpha\, \phi_a\, [N_a - n_c\, (t_{j-1})] + \beta\, [n_d\, (t_{j-1}) - n_c\, (t_{j-1})] \tag{4.155}$$

$$t_{j-1} \le t \le t$$

involving five unknowns N_a, ϕ_a, α, β, $r_c(t)$. From assumption (j), $n_d(t)$ is known exactly so that,

$$r_d(t) = \frac{dn_d(t)}{dt} \tag{4.156}$$

is known exactly.

Rushforth, Staffanson, and Crawford show through a linear system's approach that the two equations can be expressed as:

$$n_d(t_j) = n_d(t_{j-1}) - \phi_a \Delta_{t_j} + N_a \phi_a \Delta_{t_j} \tag{4.157}$$

and

$$n_c(t_j) = \beta\Delta_{t_j} n_d(t_{j-1}) + n_c(t_{j-1}) - n_c(t_{j-1}) \Delta_{t_j} [\beta + \alpha\phi_a] + \alpha N_a \phi_a \Delta_{t_j} \tag{4.158}$$

with

$$\Delta_{t_j} = t_j - t_{j-1} . \tag{4.159}$$

Here there are two equations in four unknowns N_a, ϕ_a, α, and β. If the additional assumption is made that Δ_{t_j} is a constant T (i.e., the new program versions are introduced as equally spaced points in time), then the two equations can be reexpressed as:

$$\underset{\sim}{\delta}(t_j) = \begin{pmatrix} n_d(t_j) - n_d(t_{j-1}) \\ n_c(t_j) - n_c(t_{j-1}) \end{pmatrix}_{2x1} = N_a\phi_a L^{j-1}B \tag{4.160}$$

where

$$L_{2x2} = \begin{pmatrix} 1 & -\phi T \\ \beta T & 1 - T[\beta + \phi_a\alpha] \end{pmatrix} \tag{4.161}$$

and

$$B_{2x1} = \begin{pmatrix} T \\ \alpha T \end{pmatrix} . \tag{4.162}$$

Their paper develops the estimates of the four unknowns using a least squares approach. The quantity for which the minimum is sought is:

$$\sum_{i=1}^{K} (\underset{\sim}{\delta}(t_i) - \text{observed increments})^2 \tag{4.163}$$

where K is the number of program versions,

$$\underset{\sim}{\delta}(t_i) = N_a\phi_a L^{i-1}B, \tag{4.164}$$

and the observed increments =

$$\begin{pmatrix} n_d(t_j) - n_d(t_{j-1}) \\ n_c(t_j) - n_c(t_{j-1}) \end{pmatrix} \tag{4.165}$$

$$= \begin{pmatrix} \text{difference in number detected from one version to the next} \\ \text{difference in number corrected from one version to the next} \end{pmatrix}$$

A computer program was developed to perform this nonlinear minimization process and is provided in their paper.

4.2.4 Schick-Wolverton Model

The next class of model considered was originally proposed by George Schick and Ray Wolverton.[6] Their model assumes that the hazard rate function is not only proportional to the number of errors in the program, but proportional to the amount of testing time as well. Their logic is that as testing progresses on a program, the chance of detecting errors increases because of "zeroing-in" on those sections of code in which errors lie. Specifically, their model is based on the following assumptions:

Model Assumptions

(a) The rate of error detection is proportional to the current error content of a program and to the amount of time spent in testing.

(b) All errors are equally likely to occur and are independent of each other.

(c) Each error is of the same order of severity as any other error.

(d) The software is operated in a similar manner as the anticipated operational usage.

(e) The errors are corrected instantaneously without introduction of new errors into the program.

The one major difference between these assumptions and the Jelinski-Moranda Model is assumption (a) with the error rate also being proportional to the amount of testing time.

The form of the hazard rate function is:

$$Z(X_i) = \phi\,[N - (i - 1)]X_i \quad , \tag{4.166}$$

where X_i is the amount of testing time spent between the discovery of the $(i - 1)$st error at time t_{i-1} and the ith error at time t_i. The quantity ϕ is the proportionality constant of assumption (a) and N is the total number of errors initially in the program. Figure 4-2 is a plot of this function over time. Using the relationship established earlier between the hazard rate function, the reliability function, and the MTBF, it can be seen that:

$$R(X_i) = \exp\left\{-\phi\,[N - (i - 1)]\,\frac{X_i^2}{2}\right\} \quad \text{(the Rayleigh distribution)} \tag{4.167}$$

and

$$\text{MTBF} = \int_0^\infty R(X_i)dx_i = \sqrt{\frac{\Pi}{2\phi[N - (i - 1)]}} \quad . \tag{4.168}$$

To develop the MLEs of ϕ and N, suppose errors are discovered at times $t_1, \ldots, t_n$ and suppose $X_i = t_i - t_{i-1}$. Then, from equation 4.167.

$$R(X_i) = \exp\left\{-\phi[N - (i - 1)]\,\frac{X_i^2}{2}\right\} \quad . \tag{4.169}$$

Now $f(X_i) = -R'(X_i)$ so the distribution of the time between the $(i - 1)$st and ith error is:

$$f(X_i) = \phi[N - (i - 1)]X_i \exp\left\{-\phi[N - (i - 1)]\,\frac{X_i^2}{2}\right\} \quad . \tag{4.170}$$

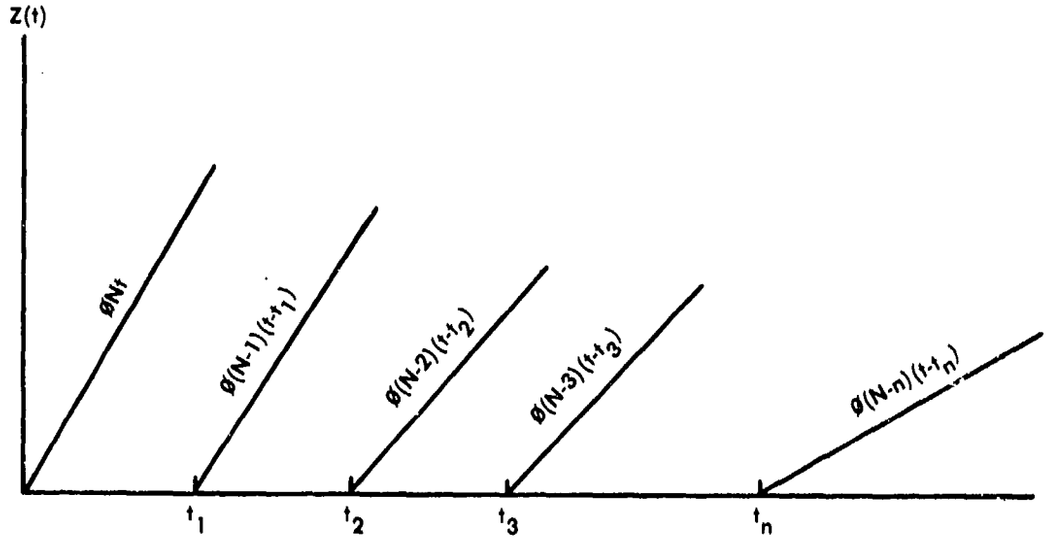

FIGURE 4-2. SCHICK-WOLVERTON HAZARD RATE FUNCTION

From assumption (b), the likelihood function based upon the X_i's is:

$$L(X_1,\ldots,X_n) = \prod_{i=1}^{n} f(X_i) = \prod_{i=1}^{n} \phi[N - (i - 1)]X_i \exp \{- \phi[N - (i - 1)] \frac{X_i^2}{2}\}. \tag{4.171}$$

Taking the partial derivatives with respect to ϕ and N of the log of the likelihood function,

$$\frac{\partial \ell nL}{\partial \phi} = \frac{n}{\phi} - \sum_{i=1}^{n} \frac{[N - (i - 1)]X_i^2}{2} \tag{4.172}$$

and

$$\frac{\partial \ell nL}{\partial N} = \sum_{i=1}^{n} \frac{1}{[N - (i - 1)]} - \phi \sum_{i=1}^{n} \frac{X_i^2}{2} \quad . \tag{4.173}$$

Setting the previous equations equal to zero, the MLEs are obtained as solutions to the following system of equations:

Estimates - Maximum Likelihood

$$\hat{\phi}_{SW} = \frac{2n}{\sum_{i=1}^{n} [\hat{N}_{SW} - (i - 1)]X_i^2} \tag{4.174}$$

and

$$\sum_{i=1}^{n} \frac{1}{[\hat{N}_{SW} - (i - 1)]} = \hat{\phi}_{SW} \frac{\sum_{i=1}^{n} X_i^2}{2} \quad . \tag{4.175}$$

The MLE of the MTBF is then given as:

$$\hat{MTBF} = \sqrt{\frac{\pi}{2\hat{\phi}_{SW}[\hat{N}_{SW} - n]}} \quad . \tag{4.176}$$

The only data requirement to implement this model is:

Data Requirement

The time of error occurrences, $(t_1,\ldots,t_n)$, or conversely, the time between error occurrences, i.e., $(X_i = t_i - t_{i-1})$.

Using the asymptotic properties of the MLEs, it can be shown (References 17 and 33) that for large n, the large sample variances are:

$$\text{var}\{\hat{\phi}_{SW}\} = \frac{\sum_{i=1}^{n} \frac{1}{(\hat{N}_{SW} - i + 1)^2}}{D_1} , \tag{4.177}$$

$$\text{var}\{\hat{N}_{SW}\} = \frac{n}{\hat{\phi}^2 D_1} , \tag{4.178}$$

$$\text{cov}(\hat{N}_{SW}, \hat{\phi}_{SW}) = \frac{-\sum_{i=1}^{n} X_i^2}{2D_1} , \tag{4.179}$$

and

$$D_1 = \left(\sum_{i=1}^{n} \frac{n}{(\hat{N}_{SW} - i + 1)^2} \right) \Big/ \hat{\phi}_{SW}^2 - \left(\sum_{i=1}^{n} \frac{X_i^2}{2} \right)^2 . \tag{4.180}$$

100 X (1 - α) percent confidence intervals can then be constructed as before as:

Confidence Intervals

$$\left(\hat{\phi}_{SW} - Z_{1-\frac{\alpha}{2}} \sqrt{\text{var}\{\hat{\phi}_{SW}\}} , \ \hat{\phi}_{SW} + Z_{1-\frac{\alpha}{2}} \sqrt{\text{var}\{\hat{\phi}_{SW}\}} \right) \tag{4.181}$$

and

$$\left(\hat{N}_{SW} - Z_{1-\frac{\alpha}{2}} \sqrt{\text{var}\{\hat{N}_{SW}\}} , \ \hat{N}_{SW} + Z_{1-\frac{\alpha}{2}} \sqrt{\text{var}\{\hat{N}_{SW}\}} \right) \tag{4.182}$$

where

$Z_{1-\frac{\alpha}{2}}$ is chosen from a standard normal table so that:

$$P\{Z \geq Z_{1-\frac{\alpha}{2}}\} = \frac{\alpha}{2} . \tag{4.183}$$

The expected time to remove the remaining N - n errors after n errors are discovered is:

Expected Time to Remove the Remaining N - n Errors

$$= \sum_{j=n+1}^{N} \sqrt{\frac{\pi}{2\phi[N - (j - 1)]}} \tag{4.184}$$

$$= \sum_{i=1}^{N-n} \sqrt{\frac{\Pi}{2\phi i}} \quad . \tag{4.185}$$

The large sample MLEs of this parameter are therefore:

Estimated Time to Remove the Remaining N - n Errors

$$= \sum_{i=1}^{\hat{N}_{SW}-n} \sqrt{\frac{\Pi}{2\hat{\phi}_{SW} i}} \quad . \tag{4.186}$$

As in the Jelinski-Moranda Model, the least squares estimates of ϕ and N are obtained by minimizing:

$$\sum_{i=1}^{n} (X_i - MTBF_i)^2 = \sum_{i=1}^{n} \left(X_i - \sqrt{\frac{\Pi}{2\phi[N - (i-1)]}} \right)^2 \quad . \tag{4.187}$$

Taking the partial derivatives with respect to ϕ and N, the least squares estimates are obtained as the solutions to the following pair of equations:

Estimates - Least Squares

$$\hat{\phi}_{LS,SW} = \frac{\frac{\Pi}{2} \left(\sum_{i=1}^{n} \frac{1}{\hat{N}_{LS,SW} - i + 1} \right)^2}{\left(\sum_{i=1}^{n} \frac{X_i}{(\hat{N}_{LS,SW} - i + 1)^{\frac{1}{2}}} \right)^2} \tag{4.188}$$

and

$$\sum_{i=1}^{n} \frac{X_i}{(\hat{N}_{LS,SW} - i + 1)^{3/2}} = \sqrt{\frac{\Pi}{2\hat{\phi}_{LS,SW}}} \sum_{i=1}^{n} \frac{1}{(\hat{N}_{LS,SW} - i + 1)^2} \quad . \tag{4.189}$$

The asymptotic variances for these estimates can be developed using the approach taken in the Jelinski-Moranda section of this paper.

For this report, only one basic extension of the Schick-Wolverton Model proposed by Lipow[38] is considered.

4.2.4.1 Lipow's Extension to the Schick-Wolverton Model. Lipow's Model uses the same assumptions as the Schick-Wolverton Model except assumption (a) of the previous paragraph is replaced by

(a) The rate of error detection is proportional to the current error content of the program and the total time previously spent in testing including an "averaged" error search time during the current time interval of testing.

Another way in which this model differs from the previous one (hence, it is not a true extension) is that it is an error count model rather than one using the "time between error occurrences." Suppose f_i errors are discovered during the ith testing interval and suppose $F_i = \sum_{j=1}^{i} f_j$ is the cumulative number of errors discovered up through i testing intervals.

Based upon the model assumptions for the hazard rate function, it can be seen that:

$$Z(x_i) = \phi[N - F_{i-1}] \left[\underline{X}_{i-1} + \frac{x_i}{2}\right] , \tag{4.190}$$

where

x_i is the amount of testing time spent between the end of testing period i - 1 and the end of the ith testing period,

and

$$\underline{X}_{i-1} = \sum_{j=1}^{i-1} x_j , \tag{4.191}$$

i.e., the cumulative amount of testing time spent through i - 1 intervals.

Since the reliability function is related to the hazard rate as:

$$R(x) = \exp\left\{-\int_0^x Z(v)dv\right\} , \tag{4.192}$$

then

$$R(x_i) = \exp\left\{\int_0^{x_i} \phi[N - F_{i-1}] \left[\underline{X}_{i-1} + \frac{v}{2}\right]dv\right\} ; \tag{4.193}$$

i.e.,

$$R(x_i) = \exp\left\{-\phi[N - F_{i-1}] \left[\underline{X}_{i-1}x_i + \frac{x_i^2}{4}\right]\right\} . \tag{4.194}$$

Also, as noted earlier,

$$MTBF = \int_0^\infty R(x_i)dx_i \tag{4.195}$$

so

$$MTBF = \int_0^\infty \exp\left\{-\phi[N - F_{i-1}]\ [\underline{X}_{i-1}x_i + \frac{x_i^2}{4}]\right\} dx_i \tag{4.196}$$

$$= \int_0^\infty \exp\left\{-\phi\ \frac{[N - F_{i-1}]}{4}\left[x_i^2 + 4\underline{X}_{i-1}x_i - 4\underline{X}_{i-1}^2 + 4\underline{X}_{i-1}^2\right]\right\}dx_i \tag{4.197}$$

$$= \int_0^\infty \exp\left\{-\phi\ \frac{[N - F_{i-1}]}{4}\left[(x_i + 2\underline{X}_{i-1})^2 - 4\underline{X}_{i-1}^2\right]\right\}dx_i \tag{4.198}$$

$$= \exp\{\phi\ [N - F_{i-1}]\ \underline{X}_{i-1}^2\} \int_0^\infty \exp\left\{-\phi\ \frac{[N - F_{i-1}]}{4}\ (x_i + 2\underline{X}_{i-1})^2\right\}dx_i \tag{4.199}$$

$$= \exp\left\{\phi\ [N - F_{i-1}]\underline{X}_{i-1}^2\right\}\sqrt{\frac{4\Pi}{\phi[N - F_{i-1}]}} \int_0^\infty \frac{1}{\sqrt{\frac{4\Pi}{\phi[N - F_{i-1}]}}}$$

$$\exp\left\{-\frac{[x_i - (-2X_{i-1})]^2}{2\left(\frac{2}{\phi[N - F_{i-1}]}\right)}\right\} dx_i \tag{4.200}$$

$$= \exp\{\phi[N - F_{i-1}]\underline{X}_{i-1}^2\}\ \frac{2\sqrt{\Pi}}{\sqrt{\phi[N - F_{i-1}]}} \cdot 1/2 \tag{4.201}$$

$$= \frac{\sqrt{\Pi}}{\sqrt{\phi[N - F_{i-1}]}}\ \exp\{\phi[N - F_{i-1}]\underline{X}_{i-1}^2\}. \tag{4.202}$$

Sukert[17] gives the MLEs for ϕ and N as the solutions to the following system of equations:

Estimates - Maximum Likelihood

$$\hat{\phi}_{LSW} = \frac{F_M}{\sum_{i=1}^{M} (\hat{N}_{LSW} - F_{i-1})x_i \ (\underline{X}_{i-1} + \frac{x_i}{2})} \tag{4.203}$$

and

$$\sum_{i=1}^{M} \frac{f_i}{\hat{N}_{LSW} - F_{i-1}} = \hat{\phi}_{LSW} \sum_{i=1}^{M} x_i \ (\underline{X}_{i-1} + \frac{x_i}{2}) \quad , \tag{4-204}$$

(Note: Sukert left out $\hat{\phi}_{LSW}$ in his equation corresponding to 4-204.)

M is the number of testing intervals and $F_M = \sum_{i=1}^{M} f_i$ is the total number of errors found in all the testing intervals. The MLE of the MTBF and the expected time to remove the remaining $N - F_M$ errors are:

$$\hat{MTBF} = \frac{\sqrt{\Pi}}{\sqrt{\hat{\phi}_{LSW}[\hat{N}_{LSW} - F_M]}} \exp\left\{ \hat{\phi}_{LSW} [\hat{N}_{LSW} - F_M]\underline{X}_M^2 \right\} \tag{4.205}$$

and

Estimated Time to Remove Remaining $N - F_M$ Errors =

$$\sum_{j=F_M}^{N-1} \frac{\sqrt{\Pi} \ \exp\left\{ \hat{\phi}_{LSW} [\hat{N}_{LSW} - F_j]\underline{X}_j^2 \right\}}{\sqrt{\hat{\phi}_{LSW}[\hat{N}_{LSW} - F_j]}} \tag{4.206}$$

Sukert also provides the large sample asymptotic variances of the estimates for ϕ and N as:

$$\text{var}\ \{\hat{\phi}_{LSW}\} = \left(\sum_{i=1}^{M} \frac{f_i}{(\hat{N}_{LSW} - F_{i-1})^2} \right) / D \quad , \tag{4.207}$$

$$\text{var}\ \{\hat{N}_{LSW}\} = \frac{F_M}{\phi_{LSW}^2 D} \quad , \tag{4.208}$$

and

$$\operatorname{cov}\left(\hat{N}_{LSW}, \hat{\phi}_{LSW}\right) = \frac{-\sum_{i=1}^{M} x_i \left(\underline{X}_{i-1} + \frac{x_i}{2}\right)}{D} \quad , \tag{4.209}$$

where

$$D = \frac{F_M}{\hat{\phi}_{LSW}^2} \left(\sum_{i=1}^{M} \frac{f_i}{(\hat{N}_{LSW} - F_{i-1})^2} \right) - \left(\sum_{i=1}^{M} x_i \left(\underline{X}_{i-1} + \frac{x_i}{2}\right) \right)^2 . \tag{4.210}$$

As in the previous paragraphs, large sample confidence intervals can be constructed for ϕ and N. From the previous formulas, it can be noticed that if:

$\underline{X}_{i-1}$ is set = 0 and for all i,

$f_i = 1$ $i = 1,\ldots,n$, so that

$F_{i-1} = i - 1$,

then the formulas for the estimates and their variances become those associated with the Schick-Wolverton Model.

4.2.5 Generalized Poisson Model

The Generalized Poisson Model (GPM) was given in a report by Schafer, Alter, Angus, and Emoto[33] for the Hughes Aircraft Company under contract to the Rome Air Development Center. Their model can be considered to be analogous in form to both the Jelinski-Moranda and Schick-Wolverton Models but taken within the error count framework. With a slight modification, it can be shown to be an extension of Lipow's Model as well. The model assumptions are given in the following.

Model Assumptions

(a) The expected number of errors occurring in any time interval is proportional to the error content at the time of testing and to some function of the amount of time spent in error testing.

(b) All errors are equally likely to occur and are independent of each other.

(c) Each error is of the same order of severity as any other error.

(d) The software is operated in a similar manner as the anticipated operational usage.

(e) The errors are corrected at the ends of the testing intervals without introduction of new errors into the program. (Note: Errors discovered in one testing interval can be corrected in others; the only restriction is that the error corrections come at the end of the testing intervals.)

Using the assumptions, their model is constructed as follows. Suppose the testing intervals are of length $x_1,\ldots,x_n$ and suppose f_i errors are discovered in the ith interval. At the end of the ith interval, a total of M_i errors are corrected. In the previous extensions of the Jelinski-Moranda and Schick-Wolverton Models, M_i was set to $\sum_{j=1}^{i} f_j$, i.e., all errors found in an interval are corrected at the end of the interval. This model relaxes this assumption.

From assumption (a),

$$E\{f_i\} = \phi[N - M_{i-1}]g_i(x_1,x_2,\ldots,x_i) \tag{4.211}$$

where ϕ is the proportionality constant, N is the initial number of errors, and g_i is some function of the amount of testing time spent previously and currently. Usually, g_i is nondecreasing with the logic that as more time is spent in testing, more errors are discovered. In the paper by Schafer, et.al., the function g_i is taken to be:

$$g_i(x_1,x_2,\ldots,x_i) = x_i^{\alpha} \quad . \tag{4.212}$$

This restriction is relaxed to show a broader class of adaptability.

For example, if

$$g_i(x_1,\ldots,x_i) = x_i \quad , \tag{4.213}$$

then the resulting formulas for the estimates are the same as the Jelinski-Moranda Model; if

$$g_i(x_1,\ldots,x_i) = x_i^2/2, \tag{4.214}$$

then the formulas are the same as the Schick-Wolverton Model; and if

$$g_i(x_1,\ldots,x_i) = x_i\left\{\sum_{j=1}^{i-1} x_j + \frac{x_i}{2}\right\} , \tag{4.215}$$

Lipow's formulas are obtained.

From assumptions (a) and (b), the joint density of the f_i's is:

$$f(f_1,\ldots,f_n) = \prod_{i=1}^{n} f(f_i) \tag{4.216}$$

$$= \prod_{i=1}^{n} \frac{[\phi(N - M_{i-1})g_i(x_1,\ldots,x_i)]^{f_i}}{f_i!} \exp\{-\phi(N - M_{i-1})g_i(x_1,\ldots,x_i)\}; \tag{4.217}$$

i.e., f_i is Poisson with mean = $\phi(N - M_{i-1})g_i$. (4.218)

Hence, the likelihood function L is:

$$L(\phi,N) = \prod_{i=1}^{n} f(f_i) \tag{4.219}$$

so that

$$\ell nL(\phi,N) = \sum_{i=1}^{n} \ell nf(f_i) \tag{4.220}$$

$$= \sum_{i=1}^{n} f_i \ell n\phi + \sum_{i=1}^{n} f_i \ell n(N - M_{i-1}) + \sum_{i=1}^{n} f_i \ell ng_i \tag{4.221}$$

$$-\phi \sum_{i=1}^{n} (N - M_{i-1})g_i - \sum_{i=1}^{n} \ell nf_i! \quad .$$

Thus, taking the partial derivatives with respect to ϕ and N, the following is obtained:

$$\frac{\partial \ell nL(\phi,N)}{\partial \phi} = \frac{\sum_{i=1}^{n} f_i}{\phi} - \sum_{i=1}^{n} (N - M_{i-1})g_i \tag{4.222}$$

and

$$\frac{\partial \ell nL(\phi,N)}{\partial N} = \sum_{i=1}^{n} \frac{f_i}{(N - M_{i-1})} - \phi \sum_{i=1}^{n} g_i \tag{4.223}$$

Thus the MLEs of ϕ and N are solutions to the following pairs of equations:

Estimates - Maximum Likelihood

$$\hat{\phi}_{GPM} = \frac{\sum_{i=1}^{n} f_i}{\hat{N}_{GPM} \sum_{i=1}^{n} g_i - \sum_{i=1}^{n} M_{i-1}g_i} \tag{4.224}$$

and

$$\sum_{i=1}^{n} \frac{f_i}{(\hat{N}_{GPM} - M_{i-1})} = \hat{\phi}_{GPM} \sum_{i=1}^{n} g_i \quad . \tag{4.225}$$

If $f_i = 1$ $i = 1,\ldots,n$, $M_{i-1} = i-1$, then $g_i = x_i$ gives the equations for the Jelinski-Moranda Model; $g_i = \frac{x_i^2}{2}$ gives the Schick-Wolverton Model equations; and and if $g_i = x_i\left\{\sum_{j=1}^{i-1} x_j + \frac{x_i}{2}\right\}$, the MLEs equations for Lipow's Model are obtained.

Following the development of the asymptotic variances of the MLEs given in the Hughes report,

$$\underset{\sim}{\hat{\delta}} = \begin{pmatrix} \hat{\phi}_{GPM} \\ \hat{N}_{GPM} \end{pmatrix} \tag{4.226}$$

has for large n, a covariance matrix that is approximately

$$\sum_{\underset{\sim}{\hat{\delta}}} \approx \begin{pmatrix} -E\left\{\frac{\partial^2 \ell nL}{\partial \phi^2}\right\} & -E\left\{\frac{\partial^2 \ell nL}{\partial \phi \partial N}\right\} \\ -E\left\{\frac{\partial^2 \ell nL}{\partial \phi \partial N}\right\} & -E\left\{\frac{\partial^2 \ell nL}{\partial N^2}\right\} \end{pmatrix}^{-1} \quad . \tag{4.227}$$

Now,

$$\frac{\partial^2 \ell nL}{\partial \phi^2} = -\sum_{i=1}^{n} \frac{f_i}{\phi^2} \tag{4.228}$$

$$\frac{\partial^2 \ell nL}{\partial \phi \partial N} = -\sum_{i=1}^{n} g_i \tag{4.229}$$

and

$$\frac{\partial^2 \ell nL}{\partial N^2} = - \sum_{i=1}^{n} \frac{f_i}{(N - M_{i-1})^2} \tag{4.230}$$

thus,

$$- E\left\{\frac{\partial^2 \ell nL}{\partial \phi^2}\right\} = \frac{1}{\phi^2} \sum_{i=1}^{n} E\{f_i\}$$

$$= \frac{1}{\phi^2} \sum_{i=1}^{n} \phi(N - M_{i-1})g_i \tag{4.231}$$

$$= \frac{1}{\phi} \sum_{i=1}^{n} (N - M_{i-1})g_i, \tag{4.232}$$

$$-E\left\{-\sum_{i=1}^{n} g_i\right\} = \sum_{i=1}^{n} g_i \tag{4.233}$$

and

$$-E\left\{\frac{\partial^2 \ell nL}{\partial N^2}\right\} = \sum_{i=1}^{n} \frac{E\{f_i\}}{(N - M_{i-1})^2} = \sum_{i=1}^{n} \frac{\phi(N - M_{i-1})g_i}{(N - M_{i-1})^2} \tag{4.234}$$

$$= \phi \sum_{i=1}^{n} \frac{g_i}{(N - M_{i-1})} \,. \tag{4.235}$$

Hence,

$$\sum_{\underset{\approx}{\hat{\delta}}} \approx \begin{pmatrix} \frac{1}{\hat{\phi}_{GPM}} \sum_{i=1}^{n} (\hat{N}_{GPM} - M_{i-1})g_i & \sum_{i=1}^{n} g_i \\ \sum_{i=1}^{n} g_i & \hat{\phi}_{GPM} \sum_{i=1}^{n} \frac{g_i}{(\hat{N}_{GPM} - M_{i-1})} \end{pmatrix}^{-1} \tag{4.236}$$

$$\approx \begin{pmatrix} \dfrac{\sum_{i=1}^{n} f_i}{\hat{\phi}^2_{GPM}} & \sum_{i=1}^{n} g_i \\ \sum_{i=1}^{n} g_i & \hat{\phi}_{GPM} \sum_{i=1}^{n} \dfrac{g_i}{(\hat{N}_{GPM} - M_{i-1})} \end{pmatrix}^{-1} \qquad (4.237)$$

Now let

$$D^* = \left(\frac{1}{\hat{\phi}^2_{GPM}} \sum_{i=1}^{n} f_i\right)\left(\hat{\phi}_{GPM} \sum_{i=1}^{n} \frac{g_i}{(\hat{N}_{GPM} - M_{i-1})}\right) - \left(\sum_{i=1}^{n} g_i\right)^2 \qquad (4.238)$$

then

$$\sum_{\hat{\underset{\sim}{\delta}}} \approx \frac{1}{D^*} \begin{pmatrix} \hat{\phi}_{GPM} \sum_{i=1}^{n} \dfrac{g_i}{(\hat{N}_{GPM} - M_{i-1})} - \sum_{i=1}^{n} g_i & \\ - \sum_{i=1}^{n} g_i & \dfrac{\sum_{i=1}^{n} f_i}{\hat{\phi}^2_{GPM}} \end{pmatrix}; \qquad (4.239)$$

i.e,

$$\text{var}\left\{\hat{\phi}_{GPM}\right\} \approx \frac{\hat{\phi}_{GPM}}{D^*} \sum_{i=1}^{n} \frac{g_i}{\left(\hat{N}_{GPM} - M_{i-1}\right)} \qquad (4\text{-}240)$$

$$\text{var}\left\{\hat{N}_{GPM}\right\} \approx \frac{\sum_{i=1}^{n} f_i}{\hat{\phi}^2_{GPM} D^*} \qquad (4.241)$$

and

$$\text{cov}\left(\hat{\phi}_{GPM}, \hat{N}_{GPM}\right) \approx -\frac{\sum_{i=1}^{n} g_i}{D^*} . \qquad (4.242)$$

Note if g_i is the appropriate function $\left\{ x_i, \frac{x_i^2}{2}, \text{ or } x_i \left[\sum_{j=1}^{i-1} x_j + \frac{x_i}{2} \right] \right\}$,

and if

$$\hat{\phi}_{GPM} \sum_{i=1}^{n} \frac{g_i}{(\hat{N}_{GPM} - M_{i-1})} = \sum_{i=1}^{n} \frac{f_i}{(\hat{N}_{GPM} - M_{i-1})^2} \quad , \tag{4.243}$$

then not only are the MLEs for the GPM the same as the previous paragraphs, but the asymptotic variances are the same as well.

From the GPM formulation, the expected number of errors in the (n + 1)st interval of testing is

$$E\left\{ f_{n+1} \right\} = \phi \ (N - M_n) g_{n+1} \ (x_1, \ldots, x_{n+1}) \quad , \tag{4.244}$$

where x_{n+1} is the anticipated testing time for the (n + 1)st session. The MLE is therefore

$$\hat{E}\left\{ f_{n+1} \right\} = \hat{\phi}_{GPM} \ (\hat{N}_{GPM} - M_n) g_{n+1} \ (x_1, \ldots, x_{n+1}). \tag{4.245}$$

Its asymptotic variance is therefore

$$\text{var}\left\{ \hat{E}\left\{ f_{n+1} \right\} \right\} \approx \left[\frac{\partial \hat{E}\left\{ f_{n+1} \right\}}{\partial \phi} \quad \frac{\partial \hat{E}\left\{ f_{n+1} \right\}}{\partial N} \right] \sum_{\hat{\underset{\sim}{\delta}}} \left[\frac{\partial \hat{E}\left\{ f_{n+1} \right\}}{\partial \phi} \quad \frac{\partial \hat{E}\left\{ f_{n+1} \right\}}{\partial N} \right]' \tag{4.246}$$

$$\approx \left[(\hat{N}_{GPM} - M_n) g_{n+1} \quad \hat{\phi}_{GPM} \ g_{n+1} \right] \sum_{\hat{\underset{\sim}{\delta}}} \left[(\hat{N}_{GPM} - M_n) g_{n+1} \quad \hat{\phi}_{GPM} \ g_{n+1} \right]' \tag{4.247}$$

$$\approx (\hat{N}_{GPM} - M_n) g_{n+1} \quad \hat{\phi}_{GPM} \ g_{n+1} \begin{pmatrix} \frac{\hat{\phi}_{GPM}}{D^*} \sum_{i=1}^{n} \frac{g_i}{(\hat{N}_{GPM} - M_{i-1})} & - \sum_{i=1}^{n} \frac{g_i}{D^*} \\ - \sum_{i=1}^{n} \frac{g_i}{D^*} & \frac{\sum_{i=1}^{n} f_i}{D^* \hat{\phi}_{GPM}^2} \end{pmatrix} \begin{pmatrix} (\hat{N}_{GPM} - M_n) \ g_{n+1} \\ \hat{\phi}_{GPM} g_{n+1} \end{pmatrix} \tag{4.248}$$

$$\approx \frac{\left(\hat{N}_{GPM} - M_n\right)^2 g_{n+1}^2 \hat{\phi}_{GPM}}{D^*} \sum_{i=1}^{n} \frac{g_i}{(\hat{N}_{GPM} - M_{i-1})}$$

$$- \frac{2\hat{\phi}_{GPM}(\hat{N}_{GPM} - M_n) g_{n+1}^2 \sum_{i=1}^{n} g_i}{D^*} + \frac{g_{n+1}^2 \sum_{i=1}^{n} f_i}{D^*} . \quad (4.249)$$

From the previous results, large sample 100 X (1 - α) percent confidence intervals for the various parameters can be constructed as:

Confidence Intervals

$$\left(\hat{\phi}_{GPM} - Z_{1-\frac{\alpha}{2}} \sqrt{\text{var}\left\{\hat{\phi}_{GPM}\right\}},\ \hat{\phi}_{GPM} + Z_{1-\frac{\alpha}{2}} \sqrt{\text{var}\left\{\hat{\phi}_{GPM}\right\}}\right)$$

$$\left(\hat{N}_{GPM} - Z_{1-\frac{\alpha}{2}} \sqrt{\text{var}\left\{\hat{N}_{GPM}\right\}},\ \hat{N}_{GPM} + Z_{1-\frac{\alpha}{2}} \sqrt{\text{var}\left\{\hat{N}_{GPM}\right\}}\right) \quad (4.250)$$

and

$$\left(\hat{E}\left\{f_{n+1}\right\} - Z_{1-\frac{\alpha}{2}} \sqrt{\text{var}\left\{\hat{E}\left\{f_{n+1}\right\}\right\}},\ \hat{E}\left\{f_{n+1}\right\} + Z_{1-\frac{\alpha}{2}} \sqrt{\text{var}\left\{\hat{E}\left\{f_{n+1}\right\}\right\}}\right) \quad (4.251)$$

where

$$P\left\{Z \geq Z_{1-\frac{\alpha}{2}}\right\} = \frac{\alpha}{2} \quad (4.252)$$

and Z is a standard normal random variable.

In the Hughes report, the least squares estimates are also derived. Those estimates are chosen to minimize:

$$S = \sum_{i=1}^{n} [f_i - \phi (N - M_{i-1})g_i]^2. \tag{4.253}$$

By taking the partial derivatives of S with respect to ϕ and N,

$$\frac{\partial S}{\partial \phi} = -2 \sum_{i=1}^{n} [f_i - \phi(N - M_{i-1})g_i] (N - M_{i-1})g_i \tag{4.254}$$

and

$$\frac{\partial S}{\partial N} = -2 \sum_{i=1}^{n} [f_i - \phi(N - M_{i-1})g_i]\phi g_i \quad . \tag{4.255}$$

The least squares estimates are then obtained as the solutions to the following system of equations:

Estimates - Least Squares

$$\hat{\phi}_{LS,GPM} = \frac{\sum_{i=1}^{n} f_i g_i}{\sum_{i=1}^{n} (\hat{N} - M_{i-1})g_i^2} \tag{4.256}$$

and

$$\sum_{i=1}^{n} f_i g_i (\hat{N}_{LS,GPM} - M_{i-1}) - \hat{\phi}_{LS,GPM} \sum_{i=1}^{n} (\hat{N}_{LS,GPM} - M_{i-1})^2 g_i^2 = 0 \; . \tag{4.257}$$

Using the large sample results for least squares estimators given in Paragraph 4.2.3 of this report and using the Hughes report, it can be shown that:

$$\hat{\underset{\sim}{\delta}}_{LS,GPM} = (\hat{\phi}_{LS,GPM} \; , \; \hat{N}_{LS,GPM})' \tag{4.258}$$

is asymptotically normally distributed with mean vector (ϕ,N) and covariance matrix:

$$\sum_{\hat{\theta}_{GPM}} \approx \frac{1}{A^2} \sum_1 \begin{pmatrix} \sum_{i=1}^{n} \hat{\phi}_{LS,GPM}(\hat{N}_{LS,GPM} - M_{i-1})^3 g_i^3 & \sum_{i=1}^{n} \hat{\phi}^2_{LS,GPM}(\hat{N}_{LS,GPM} - M_{i-1})^2 g_i^3 \\ \sum_{i=1}^{n} \hat{\phi}^2_{LS,GPM}(\hat{N}_{LS,GPM} - M_{i-1})^2 g_i^3 & \sum_{i=1}^{n} \hat{\phi}^3_{LS,GPM}(N_{LS,GPM} - M_{i-1}) g_i^3 \end{pmatrix} \sum_1 \tag{4.259}$$

where

$$\sum_1 = \begin{pmatrix} \sum_{i=1}^{n} \hat{\phi}^2_{LS,GPM} g_i^2 & -\sum_{i=1}^{n} \hat{\phi}_{LS,GPM}(\hat{N}_{LS,GPM} - M_{i-1}) g_i^2 \\ -\sum_{i=1}^{n} \hat{\phi}_{LS,GPM}(\hat{N}_{LS,GPM} - M_{i-1}) g_i^2 & \sum_{i=1}^{n} (\hat{N}_{LS,GPM} - M_{i-1})^2 g_i^2 \end{pmatrix} \tag{4-260}$$

and

$$A = \left(\sum_{i=1}^{n} \hat{\phi}^2_{LS,GPM} g_i^2 \sum_{i=1}^{n} (\hat{N}_{LS,GPM} - M_{i-1})^2 g_i^2\right) - \left(\sum_{i=1}^{n} \hat{\phi}_{LS,GPM}(\hat{N}_{LS,GPM} - M_{i-1}) g_i^2\right)^2 . \tag{4.261}$$

Large sample confidence intervals can then be constructed using these results as before.

The data requirements necessary to implement this model are:

Data Requirements

(a) The lengths of the various testing intervals, i.e., $x_1,\ldots,x_n$,

(b) The number of errors corrected at the end of each testing period, and

(c) The number of errors discovered in each interval of testing, i.e., f_i's.

In the Hughes report[33], an extensive analysis was done on the properties of the least squares and MLEs for the specific model of $g_i(x_1,\ldots,x_i) = x_i^{\alpha}$ with α a positive integer. Thus the GPM could be taken as a three parameter estimation problem of N, ϕ, and α or a two parameter problem if α is specified. The report noted problems with lack of convergence, convergence to nonoptimal solutions, and lack of uniqueness for the estimates. The report noted that the MLEs had a greater tendency to these problems than the least squares. Both procedures were dependent upon a "good" initial starting value for the estimates in order to achieve convergence. Other problems with the estimation procedures included obtaining solutions which violated model assumptions and oscillation of the estimates in the convergence process.

4.2.6 Geometric Model. This model was proposed by Moranda (References 41 and 42) and is a variation of the Jelinski-Moranda "De-Eutrophication" Model. It is an interesting model because, unlike all of the previous models discussed, it does not assume a fixed finite number of errors in the program, nor does it assume the errors are equally likely to occur. It assumes that as debugging progresses, the errors become harder to detect. By operating on the premise that a program is never completely error free (because of error introduction in the process of correcting a detected error), this model can be utilized for error analysis. The specific model assumptions are:

Model Assumptions

(a) There is an infinite number of total errors (i.e., the program is never error free),

(b) All errors do not have the same chance of detection,

(c) The detections of errors are independent,

(d) The software is operated in a similar manner as the anticipated operational usage, and

(e) The error detection rate forms a geometric progression and is constant between error occurrences.

From these assumptions, the hazard rate for this model is of the form

$$Z(t) = D\phi^{i-1} \tag{4.262}$$

for any time t between the occurrence of the (i - 1)st error and the ith. The hazard rate function is initially a constant D which decreases in a geometric progression ($0 < \phi < 1$) as error detection occurs. Figure 4-3 is a graphic representation of this hazard rate function.

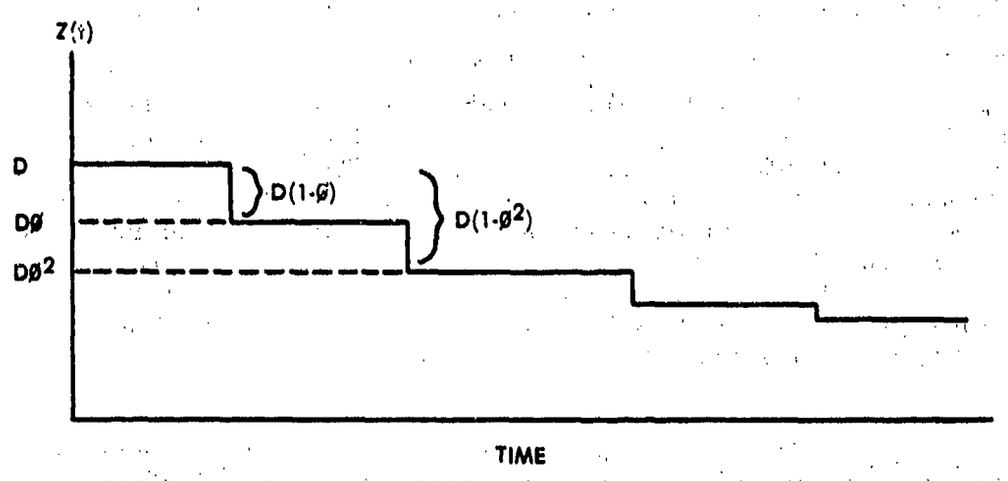

FIGURE 4-3. GEOMETRIC "DE-EUTROPHICATION" PROCESS

Notice from the graph the ratio of the change in the error detection rate, Z(t), from the discovery of the ith error to the (i + 1)st, i.e.,

$$\frac{\text{Change in } Z(t) \text{ on the discovery of the ith error}}{\text{Change in } Z(t) \text{ on the discovery of the (i+1)st error}} = \frac{D\phi^{i-1} - D\phi^{i-2}}{D\phi^{i} - D\phi^{i-1}}$$

$$= \frac{1}{\phi} > 1 \quad . \tag{4.263}$$

Thus the size of the step gets smaller as errors are discovered. This means that latter errors are more difficult to find and do not have as dramatic effect on lowering the error rate as earlier detected ones.

Again, if $X_i = t_i - t_{i-1}$ is the time of discovery between the ith and (i - 1)st error, then using assumptions (c) and (e), the X_i's are assumed independent exponentials with rate $Z(t_i)$, i.e.,

$$f(X_i) = D\phi^{i-1} \exp\{-D\phi^{i-1}X_i\} \quad . \tag{4.264}$$

The likelihood function for the X_i's is then:

$$L(X_1,\ldots,X_n) = \prod_{i=1}^{n} f(X_i) = D^n \prod_{i=1}^{n} \phi^{i-1} \exp\{-D \sum_{i=1}^{n} \phi^{i-1}X_i\} \quad . \tag{4.265}$$

Thus the log of the likelihood function is:

$$\ell nL = n\ell nD + \sum_{i=1}^{n} (i-1)\ell n\phi - D \sum_{i=1}^{n} \phi^{i-1}X_i. \tag{4.266}$$

The MLEs of D and ϕ are obtained as the solutions to the following pair of equations:

$$\frac{\partial \ell nL}{\partial D} = \frac{n}{D} - \sum_{i=1}^{n} \phi^{i-1}X_i = 0 \tag{4.267}$$

and

$$\frac{\partial \ell nL}{\partial \phi} = \sum_{i=1}^{n} \frac{(i-1)}{\phi} - D \sum_{i=1}^{n} (i-1)\phi^{i-2}X_i = 0. \tag{4.268}$$

Estimates - Maximum Likelihood

That is:

$$\hat{D}_G = \frac{\hat{\phi}_G^n}{\sum_{i=1}^{n} \hat{\phi}_G^i X_i} \tag{4.269}$$

and

$$\frac{\sum_{i=1}^{n} i\hat{\phi}_G^i X_i}{\sum_{i=1}^{n} \hat{\phi}_G^i X_i} = \frac{n+1}{2} \quad . \tag{4.270}$$

From the estimates, the MLE of the MTBF after n errors have been observed can be estimated as:

$$\hat{MTBF} = \hat{E}\{X_{n+1}\} = \frac{1}{\hat{D}_G \hat{\phi}_G^n} \quad . \tag{4.271}$$

The model cannot be used to estimate the total number of errors in the program but, it can be used to estimate the "purity" level after n errors are observed. The estimated degree of "purification" for a program is usually given by the ratio:

$$\frac{Z(t_o) - Z(t_n)}{Z(t_o)} = \frac{D - D\phi^n}{D} = 1 - \phi^n \quad , \tag{4.272}$$

the change in the hazard rate function from the beginning of testing to the end versus what it was at the beginning. The estimate of this purification level is therefore:

$$\hat{PL} = 1 - \hat{\phi}_G^n. \tag{4.273}$$

The large sample estimates for variances of these estimators are derived again using the large sample properties of the MLEs (see Paragraph 4.2.5.).

Now

$$\frac{\partial^2 \ell nL}{\partial D^2} = - \frac{n}{D^2} \quad , \tag{4.274}$$

$$\frac{\partial^2 \ell nL}{\partial \phi \partial D} = - \sum_{i=1}^{n} (i - 1)\phi^{(i-2)} X_i \quad , \tag{4.275}$$

and

$$\frac{\partial^2 \ell nL}{\partial \phi^2} = - \frac{n(n - 1)}{2\phi^2} - D \sum_{i=1}^{n-2} i(i + 1) \phi^{i-1} X_{i+2} \quad . \tag{4.276}$$

Hence,

$$E \left\{ - \frac{\partial^2 \ell nL}{\partial D^2} \right\} = \frac{n}{D^2} \tag{4.277}$$

$$E \left\{ - \frac{\partial^2 \ell nL}{\partial \phi \partial D} \right\} = \sum_{i=1}^{n} (i - 1)\phi^{i-2} E \{X_i\} \tag{4.278}$$

$$= \sum_{i=1}^{n} (i - 1)\phi^{i-2} \left(\frac{1}{D\phi^{i-1}} \right) \tag{4.279}$$

$$= \frac{1}{D\phi} \sum_{i=1}^{n} (i - 1) = \frac{n(n - 1)}{2D\phi} \tag{4.280}$$

and

$$E \left\{ - \frac{\partial^2 \ell nL}{\partial \phi^2} \right\} = \frac{n(n - 1)}{2\phi^2} + D \sum_{i=1}^{n-2} i(i + 1)\phi^{i-1} E \left\{ X_{i+2} \right\} \tag{4.281}$$

$$= \frac{n(n - 1)}{2\phi^2} + D \sum_{i=1}^{n-2} i(i + 1)\phi^{i-1} \left(\frac{1}{D\phi^{i+1}} \right) \tag{4.282}$$

$$= \frac{n(n-1)}{2\phi^2} + \frac{1}{\phi^2}\left[\frac{(n-2)(n-1)(2n-3)}{6} + \frac{(n-2)(n-1)}{2}\right] \quad (4.283)$$

$$= \frac{n(2n-1)(n-1)}{6\phi^2} \; . \quad (4.284)$$

Hence for large n,

$$\underset{\sim}{\hat{\delta}} = \begin{pmatrix} \hat{D}_G \\ \hat{\phi}_G \end{pmatrix} \quad (4.285)$$

is asymptotically bivariate normal with mean vector $(D,\phi)'$ and covariance matrix

$$\sum_{\underset{\sim}{\hat{\delta}}} \approx \begin{pmatrix} \frac{n}{D^2} & \frac{n(n-1)}{2D\phi} \\ \frac{n(n-1)}{2D\phi} & \frac{n(2n-1)(n-1)}{6\phi^2} \end{pmatrix}^{-1} \quad (4.286)$$

$$= \begin{pmatrix} \frac{2D^2(2n-1)}{n(n+1)} & \frac{-6D\phi}{n(n+1)} \\ \frac{-6D\phi}{n(n+1)} & \frac{12\phi^2}{n(n-1)(n+1)} \end{pmatrix} , \quad (4.287)$$

i.e.,

$$\text{var}\{\hat{D}_G\} \approx \frac{2\hat{D}_G(2n-1)}{n(n+1)} , \quad (4.288)$$

$$\text{var}\{\hat{\phi}_G\} \approx \frac{12\,\hat{\phi}_G^2}{n(n-1)(n+1)} , \quad (4.289)$$

and

$$\text{cov}(\hat{\phi}_G, \hat{D}_G) \approx -\frac{6\hat{D}_G\hat{\phi}_G}{n(n+1)} \; . \quad (4.290)$$

The estimated variances of the $\hat{\text{MTBF}}$ and $\hat{\text{PL}}$ are then obtained from:

$$\text{var}\{\hat{\text{MTBF}}\} \approx \left(\frac{\partial g}{\partial D} \; , \; \frac{\partial g}{\partial \phi}\right) \sum_{\underset{\sim}{\hat{\delta}}} \left(\frac{\partial g}{\partial D} \; , \; \frac{\partial g}{\partial \phi}\right)' \quad (4.291)$$

where

$$g = \frac{1}{D\phi^n} \tag{4.292}$$

and

$$\text{var}\{\hat{PL}\} \approx \left(\frac{\partial f}{\partial D} \quad \frac{\partial f}{\partial \phi}\right) \sum_{\hat{\delta}} \left(\frac{\partial f}{\partial D} \quad \frac{\partial f}{\partial \phi}\right) \tag{4.293}$$

where

$$f = 1 - \phi^n \quad . \tag{4.294}$$

These can be shown to be:

$$\text{var}\{\hat{MTBF}\} = \frac{2(2n - 1)}{D_G^2 \, \hat{\phi}_G^{2n} n(n + 1)} \tag{4.295}$$

and

$$\text{var}\{\hat{PL}\} = \frac{12n \, \hat{\phi}_G^{2n}}{(n - 1)(n + 1)} \quad . \tag{4.296}$$

As in previous paragraphs, using these large sample results, confidence intervals can be constructed for the various parameters.

In Tal's paper (Reference 32), least squares estimates of D and ϕ are derived from the times between error occurrences (X_i's) and the times of error occurrences (t_i's). Specifically, for the X_i's, minimize the following:

$$S_1(D,\phi) = \sum_{i=1}^{n} \left(X_i - \frac{1}{D\phi^{i-1}}\right)^2 \quad . \tag{4.297}$$

Taking the partial derivatives of S_1 with respect to D and ϕ, these are obtained:

$$\frac{\partial S_1}{\partial D} = 2\left(\sum_{i=1}^{n} \frac{X_i}{D^2\phi^{i-1}} - \sum_{i=1}^{n} \frac{1}{D^3\phi^{2(i-1)}}\right) \tag{4.298}$$

and

$$\frac{\partial S_1}{\partial \phi} = 2\left(\sum_{i=1}^{n} \frac{X_i(i - 1)}{D\phi^{i}} - \sum_{i=1}^{n} \frac{(i - 1)}{D^2\phi^{2i-1}}\right) \quad . \tag{4.299}$$

The least squares estimates are therefore the solutions of the pair of equations:

Estimates - Least Squares

$$\hat{D}_{LS,G} = \frac{\sum_{i=1}^{n} \frac{1}{\hat{\phi}_{LS,G}^{2(i-1)}}}{\sum_{i=1}^{n} \frac{X_i}{\hat{\phi}_{LS,G}^{i-1}}} \tag{4.300}$$

and

$$\left(\sum_{i=1}^{n} \frac{1}{\hat{\phi}_{LS,G}^{2(i-1)}}\right)\left(\sum_{i=1}^{n} \frac{X_i(i-1)}{\hat{\phi}_{LS,G}^{(i-1)}}\right) - \left(\sum_{i=1}^{n} \frac{X_i}{\hat{\phi}_{LS,G}^{(i-1)}}\right)\left(\sum_{i=1}^{n} \frac{(i-1)}{\hat{\phi}_{LS,G}^{2(i-1)}}\right) = 0. \tag{4.301}$$

The least squares estimates based upon the times of occurrences are obtained by minimizing:

$$S_2(D,\phi) = \sum_{i=1}^{n}\left(t_i - \sum_{j=1}^{i} \frac{1}{D\phi^{j-1}}\right)^2 \tag{4.302}$$

Again taking the partial derivatives of S_2 with respect to D and ϕ, the least squares estimates are obtained as the solutions to the following equations:

$$\hat{D}_{LS,G} \sum_{i=1}^{n} t_i \left(\sum_{j=1}^{i} \frac{1}{\hat{\phi}_{LS,G}^{j-1}}\right) - \sum_{i=1}^{n} \left(\sum_{j=1}^{i} \frac{1}{\hat{\phi}_{LS,G}^{j-1}}\right)^2 = 0 \tag{4.303}$$

and

$$\sum_{i=1}^{n} t_i \left(\sum_{j=1}^{i} \frac{(j-1)}{\hat{D}_{LS,G}\hat{\phi}_{LS,G}^{j}}\right) - \sum_{i=1}^{n} \left(\sum_{j=1}^{i} \frac{1}{\hat{D}_{LS,G}\hat{\phi}_{LS,G}^{j-1}}\right)\left(\sum_{j=1}^{i} \frac{(j-1)}{\hat{D}_{LS,G}\hat{\phi}_{LS,G}^{j}}\right) = 0 \tag{4.304}$$

i.e., letting

$$C_i = \sum_{j=1}^{i} \frac{1}{\hat{\phi}_{LS,G}^{j-1}} \tag{4.305}$$

and

$$B_i = \sum_{j=1}^{i} \frac{(j-1)}{\phi_{LS,G}^{*j}} \quad ; \tag{4.306}$$

the least squares estimates are the solutions of:

Estimates - Least Squares

$$\hat{D}_{LS,G} \sum_{i=1}^{n} t_i B_i - \sum_{i=1}^{n} C_i B_i = 0 \tag{4.307}$$

and

$$\hat{D}_{LS,G} \sum_{i=1}^{n} t_i C_i - \sum_{i=1}^{n} C_i^2 = 0 \quad . \tag{4.308}$$

Using the results of Paragraph 4.2.3, the asymptotic variances of these various estimates can be established. For large n, the estimates have a covariance matrix of the form:

$$\frac{1}{\Delta^2} \sum_1 \begin{pmatrix} \sum_{i=1}^{n} \sigma_i^2(\theta_1,\theta_2) \left(\frac{\partial g_i}{\partial \theta_1}\right)^2 & \sum_{i=1}^{n} \sigma_i^2(\theta_1,\theta_2) \left(\frac{\partial g_i}{\partial \theta_1}\right)\left(\frac{\partial g_i}{\partial \theta_2}\right) \\ \sum_{i=1}^{n} \sigma_i^2(\theta_1,\theta_2) \left(\frac{\partial g_i}{\partial \theta_1}\right)\left(\frac{\partial g_i}{\partial \theta_2}\right) & \sum_{i=1}^{n} \sigma_i^2(\theta_1,\theta_2) \left(\frac{\partial g_i}{\partial \theta_2}\right)^2 \end{pmatrix} \sum_1 \tag{4.309}$$

where

$$\Delta = \sum_{i=1}^{n} \left(\frac{\partial g_i}{\partial \theta_1}\right)^2 \sum_{i=1}^{n} \left(\frac{\partial g_i}{\partial \theta_2}\right)^2 - \left[\sum_{i=1}^{n} \left(\frac{\partial g_i}{\partial \theta_1}\right)\left(\frac{\partial g_i}{\partial \theta_2}\right)\right]^2 \tag{4.310}$$

and

$$\sum_1 = \begin{pmatrix} \sum_{i=1}^{n} \left(\frac{\partial g_i}{\partial \theta_2}\right)^2 & - \sum_{i=1}^{n} \frac{\partial g_i}{\partial \theta_1} \frac{\partial g_i}{\partial \theta_2} \\ - \sum_{i=1}^{n} \frac{\partial g_i}{\partial \theta_1} \frac{\partial g_i}{\partial \theta_2} & \sum_{i=1}^{n} \left(\frac{\partial g_i}{\partial \theta_1}\right)^2 \end{pmatrix} . \tag{4.311}$$

For the estimates based upon the X_i's,

$$\frac{\partial g_i}{\partial \theta_1} = \frac{\partial g_i}{\partial D} = - \frac{1}{D^2 \phi^{i-1}} \tag{4.312}$$

and

$$\frac{\partial g_i}{\partial \theta_2} = \frac{\partial g_i}{\partial \phi} = - \frac{(i - 1)}{D\phi^i} \tag{4.313}$$

since

$$g_i (\theta_1,\theta_2) = g_i (D,\phi) = \frac{1}{D\phi^{i-1}} \quad . \tag{4.314}$$

Also it is seen that,

$$\sigma_i^2 (\theta_1,\theta_2) = \text{var}\{X_i\} = \frac{1}{D^2\phi^{2(i-1)}} \, . \tag{4.315}$$

For the estimates based upon the t_i's,

$$\frac{\partial g_i}{\partial \theta_1} = \frac{\partial g_i}{\partial D} = - \frac{1}{D^2} \sum_{j=1}^{i} \frac{1}{\phi^{j-1}} \quad ; \tag{4.316}$$

$$\frac{\partial g_i}{\partial \theta_2} = \frac{\partial g_i}{\partial \phi} = - \sum_{j=1}^{i} \frac{(j - 1)}{D\phi^j} \tag{4.317}$$

with

$$g_i = \sum_{j=1}^{i} \frac{1}{D\phi^{j-1}} \tag{4.318}$$

and

$$\sigma_i^2 (\theta_1,\theta_2) = \text{var}\{t_i\} = \text{var}\left\{ \sum_{j=1}^{i} X_j \right\} = \sum_{j=1}^{i} \text{var}\{X_j\}$$

$$= \sum_{j=1}^{i} \frac{1}{D^2\phi^{2(j-1)}} \quad . \tag{4.319}$$

The only data required to implement this model are:

Data Requirement

Either the times of error occurrences, t_i's, or the time between error occurrences, X_i's.

4.2.6.1 Modified Geometric "De-Eutrophication" Model. The only extension of the Geometric Model that is considered is due to Lipow[38] and discussed in Sukert.[17] The extension is made to relax the assumption of an infinite number of errors being present in the code. The model assumptions are:

Model Assumptions

(a) All errors do not have the same chance of detection,

(b) The detections of errors are independent,

(c) The software is operated in a similar manner as the anticipated operational usage, and

(d) The error detection rate during the ith time interval of testing is:

$$Z(t) = D\phi^{n_{i-1}} \quad \text{for} \quad t_{i-1} \leq t \leq t_i \tag{4.320}$$

where D and $0 < \phi < 1$ are as in the previous paragraph and n_{i-1} is the cumulative number of errors found up to the ith interval of testing. The form that this hazard rate function takes is given in Figure 4-4.

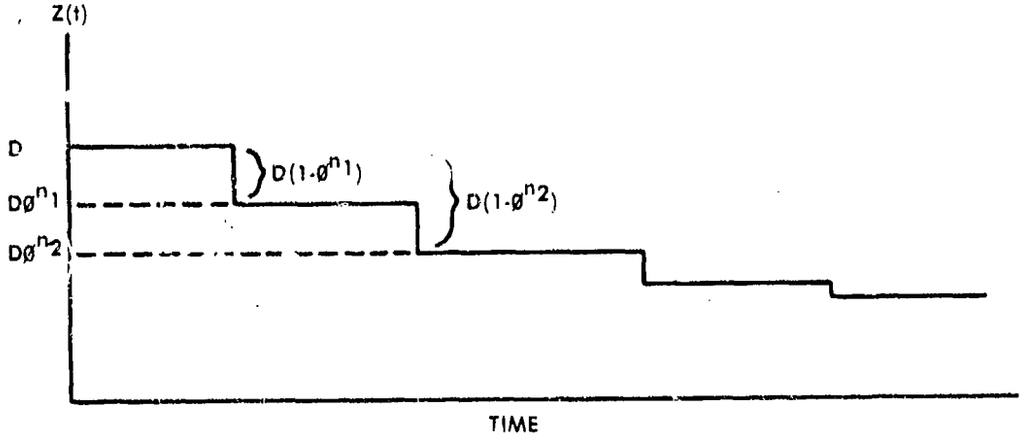

FIGURE 4-4. MODIFIED GEOMETRIC "DE-EUTROPHICATION" PROCESS

The development of the MLEs for ϕ and D proceeds in essentially the same manner as for the Geometric Model. The resulting estimates are obtained as the solutions of the following pair of equations:

Estimates - Maximum Likelihood

$$\hat{D}_{MG} = \frac{n}{\sum_{i=1}^{m} \hat{\phi}_{MG}^{n_{i-1}} X_i} \tag{4.321}$$

and

$$\frac{1}{\hat{\phi}_{MG}} \sum_{i=1}^{m} n_{i-1} = \hat{D}_{MG} \sum_{i=1}^{m} n_{i-1} \hat{\phi}_{MG}^{n_{i-1}-1} X_i \tag{4.322}$$

where m is the number of testing intervals of each length X_i, i=1,...,m and $n = \sum_{i=1}^{m} n_i$ is the total number of errors discovered. Notice that the hazard rate function and the estimates become those of the previous section when $n_{i-1} = i-1$. The MLEs of the MTBF and the reliability of the program after m intervals of testing are:

$$\text{Reliability} = \hat{R}(t) = \exp\left\{-\hat{D}_{MG}\left(\hat{\phi}_{MG}^{n_m}\right)t\right\} \quad , \quad t_m<t \tag{4.323}$$

and

$$\hat{MTBF} = \frac{1}{\hat{D}_{MG}\hat{\phi}_{MG}^{n_m}} \quad . \tag{4.324}$$

The estimated degree of "purification" for a program is obtained as in the previous paragraph as:

$$\hat{PL} = \frac{Z(t_o) - Z(t_m)}{Z(t_o)} = \frac{\hat{D}_{MG} - \hat{D}_{MG}\hat{\phi}_{MG}^{n_m}}{\hat{D}_{MG}} = 1 - \hat{\phi}_{MG}^{n_m} \quad . \tag{4.325}$$

The large sample covariance matrix of the MLEs of D and ϕ can be shown, following the same procedure as employed in the previous paragraph, to be:

$$\sum_{\underset{\sim}{\hat{\delta}}_{MG}} \approx \frac{1}{\Delta} \begin{pmatrix} \frac{m}{\hat{D}_{MG}^2} & -\frac{\sum_{i=1}^{m} n_{i-1}}{\hat{\phi}_{MG}\hat{D}_{MG}} \\ -\frac{\sum_{i=1}^{m} n_{i-1}}{\hat{\phi}_{MG}\hat{D}_{MG}} & \frac{\sum_{i=1}^{m} n_{i-1}^2}{\hat{\phi}_{MG}^2} \end{pmatrix} \tag{4.326}$$

with

$$\Delta = \frac{m}{\hat{D}_{MG}^2\hat{\phi}_{MG}^2} \sum_{i=1}^{m} n_{i-1}^2 - \left(\frac{\sum_{i=1}^{m} n_{i-1}}{\hat{\phi}_{MG}\hat{D}_{MG}}\right)^2 \tag{4.327}$$

and

$$\underset{\sim}{\hat{\delta}}_{MG} = (\hat{\phi}_{MG}, \hat{D}_{MG})' . \tag{4.328}$$

The estimated variances of the other estimates can be found by pre- and post-multiplying the covariance matrix $\sum_{\underset{\sim}{\hat{\delta}}_{MG}}$ by:

$\left(\frac{\partial f}{\partial D} \quad \frac{\partial f}{\partial \phi}\right)$, where f is any of the following functions

$$f(D,\phi) = \exp\left\{-D\,\phi^{n_m} t\right\} , \tag{4.329}$$

$$f(D,\phi) = \frac{1}{D\phi^{n_m}} , \tag{4.330}$$

or

$$f(D,\phi) = 1 - \phi^{n_m} . \tag{4.331}$$

The data required for implementation of this model are:

Data Requirements

(a) The length of the testing intervals, X_i i=1,...,m, and

(b) The number of errors detected in each interval.

4.2.7 Geometric Poisson Model

This model was also proposed by Moranda[41] as an alternative to the Geometric Model if the reporting of the software error detections is on a periodic basis. As in the previous Modified Geometric Model, only the numbers of error occurrences per testing interval are needed. Unlike the previous model, however, the testing intervals are all assumed to be the same length; e.g., a testing period is composed of a day, week, etc. Additionally, since the model assumes a constant rate of error occurrence during a time period, the model is best applied to situations in which the length of the reporting period is small in relationship to the overall length of the testing time. The model assumptions are:

Model Assumptions

(a) There is a nonfinite number of errors.

(b) The detections of errors are independent.

(c) The errors do not have the same chance of detection.

(d) The software is operated in a similar manner as the anticipated operational usage.

(e) During the ith time period, the number of errors detected, f_i, during that period follows a Poisson distribution with parameter $D\phi^{i-1}$ where D is the initial detection rate and ϕ is the constant of proportionality where $0 < \phi < 1$.

(f) Each error discovered is either corrected or not counted again.

From assumption (e), the detection rate follows a geometric progression from one testing period to the next. Initially, the detection rate is constant D. After the first reporting period, the detection rate is assumed proportional to the initial rate, i.e., it is then ϕD, and so on. The hazard rate for this model is:

$$Z(t) = D\phi^{i-1} \tag{4.332}$$

for $t_{i-1} < t < t_i$ during the ith time period. Notice how this compares to the hazard rate functions for the Geometric and Modified Geometric Models. Here the t_i's are fixed, while for the Geometric they were random.

Since the number of error detections in a reporting period follows a Poisson distribution, the likelihood function for the m reporting periods is:

$$L(f_1,\ldots,f_m) = \prod_{i=1}^{m} \left[\frac{\left(D\phi^{i-1}\right)^{f_i} \exp\left\{ - D\phi^{i-1} \right\}}{f_i!} \right] . \tag{4.333}$$

Thus

$$\ell nL = \sum_{i=1}^{m} f_i \ell nD + \sum_{i=1}^{m} f_i (i - 1)\ \ell n\phi + \sum_{i=i}^{m} -\ell n(f_i!) - \sum_{i=1}^{m} D\phi^{i-1} \quad . \quad (4.334)$$

Hence,

$$\frac{\partial \ell nL}{\partial D} = \frac{\sum_{i=1}^{m} f_i}{D} - \sum_{i=1}^{m} \phi^{i-1} \quad (4.335)$$

and

$$\frac{\partial \ell nL}{\partial \phi} = \frac{\sum_{i=1}^{m} f_i (i - 1)}{\phi} - D \sum_{i=1}^{m} (i - 1)\phi^{i-2} \quad . \quad (4.336)$$

The MLEs are then obtained as the solutions to the following pair of equations:

Estimates - Maximum Likelihood

$$\hat{D}_{GP} = \frac{\sum_{i=1}^{m} f_i}{\sum_{i=1}^{m} \hat{\phi}_{GP}^{i-1}} \quad (4.337)$$

and

$$\frac{\sum_{i=1}^{m} f_i}{\sum_{i=1}^{m} f_i (i - 1)} = \frac{\sum_{i=1}^{m} \hat{\phi}_{GP}^{i-1}}{\sum_{i=1}^{m-1} i\ \hat{\phi}_{GP}^{i}} = \frac{\left(1 - \hat{\phi}_{GP}^{m}\right)\left(1 - \hat{\phi}_{GP}\right)}{\hat{\phi}_{GP} + (m - 1)\ \hat{\phi}_{GP}^{m+1} - m\hat{\phi}_{GP}^{m}} \quad , \quad (4.338)$$

using the fact that ϕ^i forms a geometric progession, so that

$$\sum_{i=1}^{m} \phi^{i-1} = \sum_{i=0}^{m-1} \phi^{i} = \frac{1 - \phi^m}{1 - \phi} \quad (4.339)$$

and

$$\sum_{i=1}^{m-1} i\phi^{i} = \phi \sum_{i=0}^{m-1} i\phi^{i-1} = \frac{\phi(1 + (m - 1)\phi^m - m\phi^{m-1})}{(1 - \phi)^2} \quad . \quad (4.340)$$

Large sample variances of these estimates can be calculated in the usual way and are found to be:

$$\operatorname{var}\left\{\hat{D}_{GP}\right\} \approx \frac{\hat{D}_{GP}}{\Delta\hat{\phi}_{GP}^{S}}\left[\sum_{i=0}^{m-1} i\,\hat{\phi}_{GP}^{i+1} + \sum_{i=1}^{m}(i-1)(i-2)\hat{\phi}_{GP}^{i}\right] , \tag{4.341}$$

$$\operatorname{var}\left\{\hat{\phi}_{GP}\right\} \approx \frac{\sum_{i=1}^{m}\hat{\phi}_{GP}^{i-1}}{\Delta D_{GP}} , \tag{4.342}$$

and

$$\operatorname{cov}(\hat{\phi}_{GP},\hat{D}_{GP}) \approx -\frac{\sum_{i=0}^{m-1} i\,\hat{\phi}_{GP}^{i-1}}{\Delta} \tag{4.343}$$

where

$$\Delta = \frac{\sum_{i=1}^{m}\hat{\phi}_{GP}^{i-1}}{\hat{\phi}_{GP}^{3}}\left[\sum_{i=0}^{m-1} i\,\hat{\phi}_{GP}^{i+1} + \sum_{i=1}^{m}(i-1)(i-2)\hat{\phi}_{GP}^{i}\right] - \left(\sum_{i=o}^{m-1} i\,\hat{\phi}^{i-1}\right)^{2} . \tag{4.344}$$

The least squares estimates are obtained by minimizing:

$$S(\phi,D) = \sum_{i=1}^{n}(f_i - D\phi^{i-1})^2 . \tag{4.345}$$

Taking partial derivatives and setting them equal to zero, the least squares estimates are obtained as the solutions of:

Estimates - Least Squares

$$\hat{D}_{GP,LS} = \frac{\sum_{i=1}^{m} f_i\hat{\phi}_{GP,LS}^{i-1}}{\sum_{i=1}^{m}\phi_{GP,LS}^{2(i-1)}} = \frac{\left(\sum_{i=1}^{m} f_i\hat{\phi}_{GP,LS}^{i-1}\right)\left(1-\hat{\phi}_{GP,LS}\right)}{\left(1-\hat{\phi}_{GP,LS}^{2m-1}\right)} \tag{4.346}$$

and

$$\sum_{i=1}^{m}(i-1)f_i\,\hat{\phi}_{GP,LS}^{i-1} - \hat{D}_{GP,LS}\sum_{i=1}^{m}(i-1)\hat{\phi}_{GP,LS}^{2(i-1)} = 0. \tag{4.347}$$

Large sample variances can be developed for these estimates as done in previous paragraphs. The estimate of the expected number of error detections in the (m + 1)st time interval is established as either:

$$\text{Expected Number of Errors in Interval } m+1 = \hat{D}_{GP}\hat{\phi}_{GP}^{m} \tag{4.348}$$

or

$$= \hat{D}_{GP,LS}\hat{\phi}_{GP,LS}^{m} \tag{4.349}$$

4.2.8 Schneidewind's Model

Norman Schneidewind[43] proposed a generalized model which includes the Geometric Poisson as a special case. The basic philosophy of this model is that as the testing progresses over time, the error detection process changes and hence, recent error counts are usually of more use than earlier counts in predicting future error counts. Three approaches are employed in utilizing the error count data. Suppose there are m intervals of testing and f_i errors were detected in the ith interval, one of the following can be done.

(a) Utilize all of the error counts for the m intervals.

(b) Ignore the error counts completely from the first s - 1 time intervals ($2 \leq s \leq m$), and only use the data from intervals s through m.

(c) Use the cumulative error count from intervals 1 through s - 1, i.e., $F_{s-1} = \sum_{i=1}^{s-1} f_i$ and the individual errors counts from interval s through m.

Schneidewind argues that approach number 1 is applicable when one feels that the error counts from all of the intervals are useful in predicting future counts. Approach number 2 is to be used when it is felt that a significant change in the error detection process has occurred and thus only the last m - s + 1 intervals are useful in future error prediction. The last approach is an intermediate one between the two others. Here it is felt that the combined error count from the first s - 1 intervals and the individual counts from the remaining are representative of the error detection behavior for future testing intervals. The model assumptions are:

Model Assumptions

(a) The number of errors detected in one interval is independent of the error count in another.

(b) The error correction rate is proportional to the number of errors to be corrected.

(c) The software is operated in a similar manner as the anticipated operational usage.

(d) The mean number of detected errors decreases from one interval to the next.

(e) The intervals are all of the same length.

(f) The rate of error detection is proportional to the number of errors within the program at the time of test. The error detection process is assumed to be a nonhomogeneous Poisson process with an exponentially decreasing error detection rate. The rate of change is taken to be of the form

$$d_i = \alpha \exp\left\{-\beta i\right\} \tag{4.350}$$

for the ith interval where $\alpha > o$ and $\beta > o$ are the constants of the model.

From assumption (f), the cumulative mean number of errors is therefore

$$D_i = \frac{\alpha}{\beta} \left[1 - \exp\{-\beta i\}\right] \quad , \tag{4.351}$$

so that for the ith interval, the mean number of errors is

$$m_i = D_i - D_{i-1} = \frac{\alpha}{\beta} \left[\exp(-\beta(i - 1)) - \exp(-\beta i)\right]. \tag{4.352}$$

The likelihood function, assuming a Poisson process, is then developed as

$$L(f_1,\ldots,f_m) = \frac{M_{s-1}^{F_{s-1}} \exp\left\{-M_{s-1}\right\}}{F_{s-1}!} \prod_{i=s}^{m} \frac{m_i^{f_i} \exp\left\{-m_i\right\}}{f_i!} \tag{4.353}$$

where M_{s-1} is the mean number of errors in the interval 1 through s-1 with s chosen as an integer value in the range $2 \leq s \leq m$.

Using the fact that:

$$\begin{aligned} m_i &= \frac{\alpha}{\beta} \left[\exp(-\beta(i - 1)) - \exp(-\beta i)\right] \\ &= \frac{\alpha}{\beta} \exp(-\beta(i - 1)[1 - \exp(-\beta)] \end{aligned} \tag{4.354}$$

and

$$\begin{aligned} M_{s-1} &= \frac{\alpha}{\beta} \left[\exp(0) - \exp(-(s - 1)\beta)\right] \\ &= \frac{\alpha}{\beta} \left[1 - \exp(-(s - 1)\beta)\right] \quad , \end{aligned} \tag{4.355}$$

Gephart et.al.[18] established that the MLEs for α and β are then obtained as:

Estimates - Maximum Likelihood (Approach c)

$$\hat{\beta}_s = \ell n(y) \tag{4.356}$$

and

$$\hat{\alpha}_s = \frac{\left(\sum_{i=1}^{m} f_i\right)\hat{\beta}}{1 - \exp\left\{-\hat{\beta}m\right\}} \tag{4.357}$$

where y is the solution of the polynomial equation,

$$\frac{(s-1)F_{s-1}}{y^{s-1}-1} + \frac{F_{s,m}}{y-1} - \frac{mF_m}{y^m - 1} = A \tag{4.358}$$

which can be simplified to:

$$Ay^{s+m} - (A + F_{s,m})y^{s+m-1} - (A + sF_{s-1} - F_{s-1})y^{m+1}$$

$$+ (A + F_{s,m} + sF_{s-1} - F_{s-1})y^m - (A - mF_m)y^s +$$

$$(A + F_{s,m} - mF_m)y^{s-1} + (A + sF_{s-1} - mF_m - F_{s-1})y$$

$$- (A + sF_{s-1} + F_{s,m} - mF_m - F_{s-1}) = 0 \quad \text{for } y > 1 \tag{4.359}$$

with

$$A = \sum_{i=0}^{m-s} (s + i - 1)f_{s+i} , \tag{4.360}$$

$$F_{s,m} = \sum_{i=s}^{m} f_i . \tag{4.361}$$

If s is set to 1 and $F_{s,m} = F_m = \sum_{i=1}^{m} f_i$, then the polynomial in equation (4.358) gives:

$$\frac{F_m}{y-1} - \frac{mF_m}{y^m - 1} = A. \tag{4.362}$$

This simplifies to:

$$Ay^{m+1} - (A + F_m)y^m + (mF_m - A)y + (A + F_m - mF_m) = 0, \tag{4.363}$$

for $y > 1$.

The MLEs derived under these conditions are for approach (a) where all error count data are used.

In equation (4.363), if $m - s + 1$ is substituted for m and the subscript of the f_i's is modified in the expression for the summations to make f_s the first error count, the MLEs for approach (b) are obtained where the first $s - 1$ error counts are ignored. For this case, the MLEs are:

$$\hat{\beta}_s = \ell n(y) \tag{4.364}$$

and

$$\hat{\alpha}_s = \frac{\left(\sum_{i=s}^{m} f_i\right)\hat{\beta}}{1 - \exp\{-\hat{\beta}m\}} \tag{4.365}$$

where y is the solution of the equation

$$Ay^{m-s+2} - (A + F_{s,m})y^{m-s+1} + ((m - s + 1)F_{s,m} - A)y + (A + F_{s,m} - (m - s + 1)F_{s,m}) = 0 \tag{4.366}$$

for $y > 1$ where

$$A = \sum_{i=0}^{m-s} i f_{s+i}. \tag{4.367}$$

From the MLEs, various other parameters can be estimated as seen in the following.

Expected Number of Errors in the $(m + 1)$st interval of testing

$$= m_{i+1} = \frac{\alpha}{\beta}\left[\exp\{-\beta i\} - \exp\{-\beta(i + 1)\}\right] \quad ; \tag{4.368}$$

Time to detect a total number of M errors

$$= \log\left[\alpha/(\alpha - \beta M)\right]/\beta \quad ; \tag{4.369}$$

and

the correction rate for the ith interval

$$= \alpha \exp\{-\beta(i - \Delta i)\} \quad ; \tag{4.370}$$

where Δi is the lag time between the detection of errors and their correction, i.e, the time to correct $D_i - C_i$ errors where C_i is the cumulative number of errors corrected up through the ith interval.

All of these parameters are estimated by substituting the appropriate MLEs for α and β. If the lag Δi is unknown, it can be estimated by finding a value for Δi such that

$$C_i = D_{i-\Delta i} \quad , \; i \geq \Delta i \quad , \tag{4.371}$$

using the empirical data.

If approach (b) or (c) is used, a determination for s needs to be made. Schneidewind suggests letting s = 2,...,m and finding the MLEs for each value of s. For each pair of estimates, the computed sum of weighted squared deviations between the error estimates m_i and the observed counts f_i for all i is computed and the one yielding the smallest sum is the chosen s. The weighted sum is given as:

$$SDW = \sum_{i=1}^{m} \exp(\beta i)\left[\left\{\frac{\alpha}{\beta}\exp(-\beta i)\right\}\left\{\exp(\beta) - 1\right\} - f_i\right]^2 . \tag{4.372}$$

Schneidewind also suggests that to decide among which of the three approaches to use [(a), (b), or (c)], the unweighted sum of squares

$$SDU = \sum_{i=m+1}^{M} \left[\frac{\alpha}{\beta}\exp(-\beta i)\left\{\exp(\beta) - 1\right\} - f_i\right]^2 \tag{4.373}$$

is computed for each approach (i.e., α and β are replaced by their estimates for the respective model). M is some specified future time. The unweighted sum of squares is calculated between the observed counts and the expected counts over the next M - m intervals. The approach yielding the smallest sum and hence, yielding the smallest differences between predicted and actual values is the one chosen.

Gephart et.al.[18] show that the models under approaches (a) and (c) (with s = 2) are equivalent to the Geometric Poisson of Paragraph 4.2.7. If

$$\alpha = \frac{D(-\ell n\phi)}{1 - \phi} \tag{4.374}$$

and

$$\beta = -\ell n\phi \quad , \tag{4.375}$$

and these are substituted into Schneidewind's Model, it becomes the Geometric Poisson Model where D and ϕ are the parameters of that model. In an equivalent manner, if

$$D = \frac{\alpha}{\beta} \left[1 - e^{-\beta}\right] \tag{4.376}$$

and

$$\phi = e^{-\beta} \quad , \tag{4.377}$$

and they are substituted into the Geometric Poisson, it becomes Schneidewind's Model.

The data required to implement any one of three models are:

Data Requirement

The error counts for each of the m intervals of testing.

4.2.9 Nonhomogeneous Poisson Process

The Nonhomogeneous Poisson Process (NHPP) Model was proposed by Amrit Goel and Kazu Okumoto (References 44, 45, and 46). Following other models that have been considered (see Paragraphs 4.2.3.2, 4.2.5, 4.2.7, and 4.2.8), this model assumes that the error counts over nonoverlapping time intervals follow a Poisson distribution. The expected number of errors for the Poisson process in an interval of time is assumed proportional to the remaining number of errors in the program at that time. Specifically, the model assumptions are as seen in the following.

Model Assumptions

(a) The software is operated in a similar manner as the anticipated operational usage.

(b) The numbers of errors, $(f_1, f_2, \ldots, f_m)$, detected in each of the respective time intervals $[(0,t_1),(t_1,t_2),(t_2,t_3),\ldots,(t_{i-1},t_i),\ldots,(t_{m-1},t_m)]$ are independent for any finite collection of times $t_1 < t_2 <, \ldots, <t_m$.

(c) Every error has the same chance of being detected and is of the same severity as any other error.

(d) The cumulative number of errors detected at any time t, (N(t)), follows a Poisson distribution with mean m(t). The mean m(t) is such that the expected number of error occurrences for any time (t,t + Δt) is proportional to the expected number of undetected errors at time t.

(e) The expected cumulative number of errors function, m(t), is assumed to be a bounded, nondecreasing function of t with

$$m(t) = 0 \qquad t = 0$$

$$m(t) = a \qquad t \to \infty$$

where a is the expected total number of errors to be eventually detected in the testing process.

Note that $f_i = N(t_i) - N(t_{i-1})$. The NHPP differs from some of the other Poisson Models considered in that this model treats the initial error content of a program as a random variable while some of the others assume it is a fixed constant. Also the time between the (i - 1)st failure and the ith failure depends upon the time to failure of the (i - 1)st rather than being independent of it.

From assumptions (d) and (e), for any time period (t,t + Δt)

$$m(t + \Delta t) - m(t) = b\{a - m(t)\}\Delta t + O(\Delta t) \tag{4.378}$$

where b is the constant of proportionality and $\frac{O(\Delta t)}{\Delta t} \to 0$ as $\Delta t \to 0$. By letting $\Delta t \to 0$, the mean function satisfies the differential equation

$$m'(t) = ab - bm(t). \tag{4.379}$$

Under the initial condition m(0) = 0, the mean function is

$$m(t) = a(1 - e^{-bt}) \ . \tag{4.380}$$

Thus,

$$\Pr\{N(t) = n\} = \frac{[m(t)]^n e^{-m(t)}}{n!}$$

with

$$m(t) = a(1 - e^{-bt}) \ . \tag{4.381}$$

For $f_i = N(t_i) - N(t_{i-1})$ and the error counts being independent, the likelihood function is therefore:

$$L(f_1,\ldots,f_m) = \prod_{i=1}^{m} \frac{[m(t_i) - m(t_{i-1})]^{f_i} \exp\{m(t_{i-1}) - m(t_i)\}}{f_i!} \tag{4.382}$$

$$= \prod_{i=1}^{m} \frac{[a(e^{-bt_{i-1}} - e^{-bt_i})]^{f_i} \exp\{a(e^{-bt_i} - e^{-bt_{i-1}})\}}{f_i!} \quad (4.383)$$

Hence,

$$\ell nL(f_1,\ldots,f_m) = \sum_{i=1}^{m} f_i \ell na + \sum_{i=1}^{m} f_i \ell n(e^{-bt_{i-1}} - e^{-bt_i})$$

$$+ a \sum_{i=1}^{m} (e^{-bt_i} - e^{-bt_{i-1}}) - \sum_{i=1}^{m} \ell nf_i! \quad . \quad (4.384)$$

Thus,

$$\frac{\partial \ell nL}{\partial a} = \frac{\sum_{i=1}^{m} f_i}{a} - (1 - e^{-bt_m}) \quad (4.385)$$

and

$$\frac{\partial \ell nL}{\partial b} = \sum_{i=1}^{m} \frac{f_i\, t_i e^{-bt_i} - t_{i-1} e^{-bt_{i-1}}}{e^{-bt_{i-1}} - e^{-bt_i}} - at_m e^{-bt_m} . \quad (4.386)$$

Thus, the MLEs are the solutions to the following system of equations:

Estimates - Maximum Likelihood

$$\hat{a}_{NHPP} = \frac{\sum_{i=1}^{m} f_i}{(1 - e^{-\hat{b}_{NHPP} t_m})} \quad (4.387)$$

and

$$\frac{t_m e^{-\hat{b}_{NHPP} t_m} \sum_{i=1}^{m} f_i}{(1 - e^{-\hat{b}_{NHPP} t_m})} = \sum_{i=1}^{m} \frac{f_i \left(t_i e^{-\hat{b}_{NHPP} t_i} - t_{i-1} e^{-\hat{b}_{NHPP} t_{i-1}} \right)}{e^{-\hat{b}_{NHPP} t_{i-1}} - e^{-\hat{b}_{NHPP} t_i}} . \quad (4.388)$$

The expected number of error detections in the next (m + 1)st interval of testing is then estimated to be:

$$\hat{m}(t_{m+1}) - \hat{m}(t_m) = \hat{a}_{NHPP}\left(e^{-\hat{b}_{NHPP}t_m} - e^{-\hat{b}_{NHPP}t_{m+1}}\right). \tag{4.389}$$

As has been done in previous paragraphs, the least squares estimates of a and b are derived by minimizing:

$$S = \sum_{i=1}^{m} \{f_i - [m(t_i) - m(t_{i-1})]\}^2 \tag{4.390}$$

$$= \sum_{i=1}^{m} \{f_i - [a\,(e^{-bt_{i-1}} - e^{-bt_i})]\}^2 \quad . \tag{4.391}$$

Thus,

$$\frac{\partial S}{\partial a} = -2 \sum_{i=1}^{m} \{f_i - [a\,(e^{-bt_{i-1}} - e^{-bt_i})]\}\left(e^{-bt_{i-1}} - e^{-bt_i}\right) \tag{4.392}$$

and

$$\frac{\partial S}{\partial b} = -2 \sum_{i=1}^{m} \{f_i - [a(e^{-bt_{i-1}} - e^{-bt_i})][a\,(t_i e^{-bt_i} - t_{i-1}e^{-bt_{i-1}})]\}. \tag{4.393}$$

The least squares estimates are therefore the solutions of this pair of equations:

Estimates - Least Squares

$$\hat{a}_{NHPP,LS} = \frac{\sum_{i=1}^{m} f_i\left(e^{-\hat{b}_{NHPP,LS}t_{i-1}} - e^{-\hat{b}_{NHPP,LS}t_i}\right)}{\sum_{i=1}^{m}\left(e^{-\hat{b}_{NHPP,LS}t_{i-1}} - e^{-b_{NHPP,LS}t_i}\right)^2} \tag{4.394}$$

and

$$\sum_{i=1}^{m} f_i \left(t_i e^{-\hat{b}_{NHPP,LS} t_i} - t_{i-1} e^{-\hat{b}_{NHPP,LS} t_{i-1}} \right)$$

$$= \hat{a}_{NHPP,LS} \sum_{i=1}^{m} \left[\left(e^{-\hat{b}_{NHPP,LS} t_{i-1}} - e^{-\hat{b}_{NHPP,LS} t_{i-1}} \right) \left(t_i e^{-\hat{b}_{NHPP,LS} t_i} - t_{i-1} e^{-\hat{b}_{NHPP,LS} t_{i-1}} \right) \right] . \quad (4.395)$$

In the report by Schafer[33], the large sample variances and covariances are derived for the MLEs and the least squares estimates.* Provided a is large, the variances and the covariances of the MLEs are:

$$\text{var}\{\hat{a}\}_{NHPP} \approx \frac{\hat{a}}{\Delta} \left(\sum_{i=1}^{m} \frac{(t_i - t_{i-1})^2 \exp(-\hat{b}(t_i + t_{i-1}))}{\left(e^{-\hat{b}t_{i-1}} - e^{-\hat{b}t_i} \right)} - t_m^2 e^{-\hat{b}t_m} \right) \quad (4.396)$$

$$\text{var}\{\hat{b}\}_{NHPP} \approx \frac{1}{\Delta} \frac{(1 - e^{-\hat{b}t_m})}{\hat{a}} , \quad (4.397)$$

and

$$\text{cov}(\hat{a}_{NHPP}, \hat{b}_{NHPP}) \approx \frac{1}{\Delta} (-t_m e^{-\hat{b}t_m}) , \quad (4.398)$$

with

$$\Delta = (1 - e^{-\hat{b}t_m}) \left(\sum_{i=1}^{m} \frac{(t_i - t_{i-1})^2 e^{-\hat{b}(t_i + t_{i-1})}}{\left(e^{-\hat{b}t_{i-1}} - e^{-\hat{b}t_i} \right)} - t_m^2 e^{-\hat{b}t_m} \right) - t_m^2 e^{-2\hat{b}t_m} \quad (4.399)$$

For the least squares estimates, the large sample variances and covariances are given in the covariance matrix:

$$\sum_{\hat{\delta}_{NHPP,LS}} \approx \frac{1}{\Delta^2} \sum_{1} \begin{pmatrix} A & B \\ B & C \end{pmatrix} \sum_{1} \quad (4.400)$$

* The author would like to express his thanks to Mr. Vijaya K. Srivastava who pointed out the errors in the original formulas and provided the correct ones.

where

$$\hat{\delta}_{NHPP,LS} = \begin{pmatrix} \hat{a}_{NHPP,LS} \\ \hat{b}_{NHPP,LS} \end{pmatrix}, \tag{4.401}$$

$$A = \sum_{i=1}^{m} a\left(e^{-bt_{i-1}} - e^{-bt_i}\right)^3, \tag{4.402}$$

$$B = \sum_{i=1}^{m} a^2\left(e^{-bt_{i-1}} - e^{-bt_i}\right)^2\left(t_i e^{-bt_i} - t_{i-1}e^{-bt_{i-1}}\right), \tag{4.403}$$

$$C = \sum_{i=1}^{m} a^3\left(e^{-bt_{i-1}} - e^{-bt_i}\right)\left(t_i e^{-bt_{i-1}} - t_{i-1}e^{-bt_{i-1}}\right)^2, \tag{4.404}$$

$$\Delta = \sum_{i=1}^{m}\left[a\left(t_i e^{-bt_i} - t_{i-1}e^{-bt_{i-1}}\right)\right]^2 \sum_{i=1}^{m}\left(e^{-bt_{i-1}} - e^{-bt_i}\right)^2 - \left(\sum_{i=1}^{m} a\left(t_i e^{-bt_i} - t_{i-1}e^{-bt_{i-1}}\right)\left(e^{-bt_{i-1}} - e^{-bt_i}\right)\right)^2, \tag{4.405}$$

and

$$\sum_1 = \begin{pmatrix} \sum_{i=1}^{m}\left[a\left(t_i e^{-bt_i} - t_{i-1}e^{-bt_{i-1}}\right)\right]^2 & -\sum_{i=1}^{m} a\left(t_i e^{-bt_i} - t_{i-1}e^{-bt_{i-1}}\right)\left(e^{-bt_{i-1}} - e^{-bt_i}\right) \\ -\sum_{i=1}^{m} a\left(t_i e^{-bt_i} - t_{i-1}e^{-bt_{i-1}}\right)\left(e^{-bt_{i-1}} - e^{-bt_i}\right) & \sum_{i=1}^{m}\left(e^{-bt_{i-1}} - e^{-bt_i}\right)^2 \end{pmatrix} \tag{4.406}$$

This large sample result is derived utilizing the results of Paragraph 4.2.3 with:

$$g_i = a\left(e^{-bt_{i-1}} - e^{-bt_i}\right) \tag{4.407}$$

and

$$\sigma_i^2 = a\left(e^{-bt_{i-1}} - e^{-bt_i}\right). \tag{4.408}$$

Equations (4.396 through 4.406) can then be used to construct large sample confidence intervals for the parameters by replacing any unknown in the variances with their respective estimates.

Goel and Okumoto[44] derive MLEs for a and b based upon the individual times of error occurrences. If S_i represents the failure time of the ith error, they show in their report that the MLEs are the solutions to the following system of equations:

$$\hat{a}_{GO} = \frac{n}{1 - \exp(-\hat{b}_{GO} s_n)} \tag{4.409}$$

and

$$\frac{n}{\hat{b}_{GO}} = \sum_{k=1}^{n} s_k + \hat{a}_{GO} s_n \exp(-\hat{b}_{GO} s_n) \tag{4.410}$$

where n is the total number of errors detected.

Using this formulation, Goel and Okumoto establish that if $S_n = s$ is the time of the last failure, then the conditional reliability function of X_{n+1} (the time between the nth and (n + 1)st failures) is given by:

$$R_{X_{n+1}}(x \mid S_n = s) = P\{X_{n+1} \geq x \mid S_n = s\} \tag{4.411}$$

$$= P\{\text{software is operational for at least x amount of time given s amount of testing}\} \tag{4.412}$$

$$= \exp\left[-a\{e^{-bs} - e^{-b(s+x)}\}\right] . \tag{4.413}$$

Okumoto and Goel[45] utilize this reliability to determine an optimal release time for a software program. Testing can continue until the desired reliability $R = R_{X_{n+1}}(x \mid S_n = s)$ is achieved for a specified operational time of x or the required testing time s can be determined for a desired reliability for a specified operational time. If

$$R = \exp[-a\{e^{-bs} - e^{-b(s+x)}\}] \tag{4.414}$$

$$= \exp[-m(x)e^{-bs}] \quad , \tag{4.415}$$

with

$$m(x) = a(1 - e^{-bx}) \tag{4.416}$$

then the desired testing time to achieve the specified R and x is:

$$s = \frac{1}{b}\left[\ln\left(m(x)\right) - \left(\ln\left(\ln\left(\frac{1}{R}\right)\right)\right)\right]. \tag{4.417}$$

Estimates based upon previous testing data are used for a and b.

In their paper, Okumoto and Goel also determine an optimum release time based upon cost considerations. Suppose:

C_1 = cost of fixing an error during testing,

C_2 = cost of fixing an error during operational use ($C_2 > C_1$),

C_3 = cost of testing per unit time,

t = software life cycle length, and

T = software release time for testing.

Since $m(t)$ is the cumulative expected number of errors in the interval $(0,t)$, then the total expected cost is:

$$C(T) = C_1 m(T) + C_2 [m(t) - m(T)] + C_3 T. \tag{4.418}$$

Differentiating the expression with respect to T, then

$$C'(T) = C_1 m'(T) - C_2 m'(T) + C_3 \tag{4.419}$$

where

$$m'(T) = abe^{-bT} . \tag{4.420}$$

Setting the right-hand side of this equation equal to zero, the following is obtained:

$$abe^{-bT} = \frac{C_3}{C_2 - C_1} . \tag{4.421}$$

Okumoto and Goel establish in their paper (Reference 45) that:

(a) If $ab > \frac{C_3}{C_2 - C_1}$, then there exists a unique feasible solution to equation (4.421) and the optimum release time is:

$$T^* = \min\{T_0, t\} \tag{4.422}$$

where

$$T_0 = \frac{1}{b} \ln\left(\frac{ab(C_2-C_1)}{C_3}\right) \tag{4.423}$$

while

(b) If $ab \leq \frac{C_3}{C_2 - C_1}$ (4.424)

then

$$T^* = 0. \tag{4.425}$$

The last comment concerning this model is that if the testing intervals are all of the same length, say T, then this model is equivalent to the Geometric-Poisson Model and hence, Schneidewind's Model with s = 1 of Paragraphs 4.2.7 and 4.2.8 respectively. If all of the testing intervals are of the same length, then the time of the ending of the ith testing interval is $t_i = iT$. The joint density of the f_i's then becomes:

$$f(f_1,\ldots,f_m) = \prod_{i=1}^{m} \frac{\left[m(t_i) - m(t_{i-1})\right]^{f_i} \exp\left\{m(t_{i-1}) - m(t_i)\right\}}{f_i!} \tag{4.426}$$

$$= \prod_{i=1}^{m} \frac{\left[a(e^{-bt_{i-1}} - e^{-bt_i})\right]^{f_i} \exp\left\{a\left(e^{-bt_i} - e^{-bt_{i-1}}\right)\right\}}{f_i!} \tag{4.427}$$

$$= \prod_{i=1}^{m} \frac{\left[a(e^{-b(i-1)T} - e^{-biT})\right] \exp\left\{a\left(e^{-ibT} - e^{-b(i-1)T}\right)\right\}}{f_i!} \tag{4.428}$$

$$= \prod_{i-1}^{m} \frac{\left[a(1-e^{-bT})e^{-b(i-1)T}\right] \exp\left[-a(1 - e^{-bT})e^{-b(i-1)T}\right]}{f_i!} \tag{4.429}$$

Notice that if:

$$a = \frac{D}{1 - \phi} \tag{4.430}$$

and

$$b = \frac{-\ell n\ \phi}{T} \tag{4.431}$$

where D and ϕ are defined in Paragraph 4.2.7, then the joint density function becomes:

$$\prod_{i=1}^{m} \frac{\left[D\phi^{i-1}\right]^{f_i} \exp\left\{-D\phi^{i-1}\right\}}{f_i!}, \tag{4.432}$$

the Geometric Poisson. Likewise, since the Geometric Poisson and Schneidewind's Model with s = 1 are equivalent, utilizing the relationships established between the two models in the last paragraph,

$$a = \frac{\alpha}{\beta} \tag{4.433}$$

and

$$b = \frac{\beta}{T} \tag{4.434}$$

are found to be the relationship between Schneidewind's and the NHPP Models.

The data required to implement this are:

Data Requirements

(a) The error counts in each of the testing intervals, (i.e., the f_i's).

(b) The times the testing intervals end, (i.e., the t_i's).

(c) The time of error occurrences (i.e., the s_i's) if an optimal release time is desired.

4.2.10 Duane's Model

The next model considered also employs a nonhomogeneous Poisson process for the error counts. This model was originally proposed by J. T. Duane[47] as a hardware reliability growth model. Duane observed that the cumulative failure rate versus cumulative testing time when plotted on ℓn-ℓn paper tended to follow a straight line for a number of systems developed at General Electric. This model has been applied with some success to software reliability modeling by Evaluation Associates, Inc., (References 48 through 50). The specific assumptions for this model are given in the following.

Model Assumptions

(a) The software is operated in a similar manner as the anticipated operational usage,

(b) Every error has the same chance of being detected and is of the same severity as any other error,

(c) The error occurrences are independent, and

(d) The cumulative number of errors detected at any time t, [N(t)], follows a Poisson distribution with mean m(t). The mean function is taken to be of the form $m(t) = \lambda t^{\beta}$.

From the assumptions, it can be seen that if

$$\frac{m(t)}{t} = \frac{\lambda t^{\beta}}{t} = \frac{\text{Expected number of errors by time t}}{\text{total testing time}} \tag{4.435}$$

is plotted on ℓn - ℓn paper versus time, or conversely, if

$$Y = \ell n \frac{m(t)}{t} = \ell n \frac{\lambda t^{\beta}}{t} = \ell n\lambda + (\beta - 1)\ell n\, t \tag{4.436}$$

is plotted on regular paper versus $\ell n(t)$, a linear relationship relating the two is obtained. That is,

$$Y = a + bX \tag{4.437}$$

with $a = \ell n\lambda$, $b = \beta - 1$, and $X = \ell n(t)$ for the latter case.

The rate at which errors are occurring is:

$$\frac{dm(t)}{dt} = \lambda\beta t^{\beta-1}. \tag{4.438}$$

Hence, the MTBF is $\frac{1}{\lambda\beta t^{\beta-1}}$. Also

if $\beta > 1$, there is no improvement in the software as time progresses. Crow[51] shows the MLEs for λ and β are:

Estimates - Maximum Likelihood

$$\hat{\lambda}_D = \frac{n}{t_n^{\beta_D}} \tag{4.439}$$

and

$$\hat{\beta}_D = \frac{n}{\sum_{i=1}^{n-1} \ell n(t_n/t_i)}, \tag{4.440}$$

where the t_i's are the observed failure times and n is the number of software errors detected.

The MLE of the MTBF for the $(m + 1)$st error occurrence is then:

$$\hat{MTBF} = \left[\hat{\lambda}_D\hat{\beta}_D t_n^{\hat{\beta}_D-1}\right]^{-1} = \frac{t_n}{n\beta_D}. \tag{4.441}$$

Crow[51] also provides a table of

$$P\left\{ MTBF/\hat{MTBF} \leq C_\alpha \right\} = \alpha \tag{4.442}$$

which can be used to construct a $100X(1 - \alpha)$ percent confidence interval for the MTBF.

Least squares estimates for $a = \ell n\lambda$ and $b = \beta - 1$ for the equation

$$\ell n \left(\frac{\text{Expected number of errors by time t}}{\text{total testing time}} \right) = a + b\ell n(t) \tag{4.443}$$

can be achieved in the standard manner as:

Estimates - Least Squares

$$\hat{a}_{D,LS} = \bar{Y} - \hat{b}_{D,LS}\bar{X} \tag{4.444}$$

and

$$\hat{b}_{D,LS} = \frac{n \sum_{i=1}^{n} X_i Y_i - \sum_{i=1}^{n} X_i \sum_{i=1}^{n} Y_i}{n \sum_{i=1}^{n} X_i^2 - \left(\sum_{i=1}^{n} X_i \right)^2} \tag{4.445}$$

where

$$X_i = \ell n(t_i) \tag{4.446}$$

and

$$Y_i = \ell n\left(\frac{i}{t_i}\right) . \tag{4.447}$$

Various confidence intervals for the parameters of the linear model can be constructed in the usual way.

The only data required to implement this model are:

Data Requirement

The times of error occurrences.

4.2.11 Execution Time Model

The next model considred is one that has been applied to the greatest number of software development programs. This is a model developed by John Musa of Bell Laboratories.[52,53,54,55,56] The interesting aspect of this model is that it is based upon the amount of CPU time involved in testing rather than on calendar (wall clock) time; but, the model attempts to relate the two. By doing this, Musa is able to model the amount of limiting resources (failure identification personnel, failure correction personnel, and computer time) that may come into play during various time segments of testing. In addition, this model eliminates the need for developing an error correction model since the error correction rate is directly related to the instantaneous failure rate during testing. The specific assumptions for this model are given in the following.

Model Assumptions

(a) The software is operated in a similar manner as the anticipated operational usage.

(b) The detections of errors are independent.

(c) All software failures are observed.

(d) The execution times between failures are piece-wise exponentially distributed (i.e., the hazard rate is a constant that changes only at each error correction).

(e) The hazard rate is proportional to the number of errors remaining in the program.

(f) The fault correction rate is proportional to the failure occurrence rate.

(g) The quantities of the resources (failure-identification personnel, failure correction personnel, and computer times) that are available are constant over a testing segment.

(h) Resource expenditures for the kth resource, $\Delta\chi_k$, associated with a change in MTBF from T_1 to T_2 can be approximated by:

$$\Delta\chi_k \simeq \theta_k \Delta\tau + \mu_k \Delta m \qquad (4.448)$$

where $\Delta\tau$ is the increment of execution time, Δm is the increment of failures experienced, θ_k is an execution time coefficient of resource expenditure, and μ_k is a failure coefficient of resource expenditure.

(i) Failure-identification personnel can be fully utilized and computer utilization is constant.

(j) Failure-correction personnel utilization is established by limitation of error queue length for any debugger. Error queue length is determined by assuming that error correction is a Poisson process and that servers are randomly assigned in time.

Assumptions (g) through (j) are needed if there is interest in modeling resource allocation for the testing segments. Only (a) through (f) are needed for reliability modeling. In fact, (a) through (e) are assumptions which are incorporated into many of the models presented in this report. Later in Paragraph 4.2.11, an equivalence relationship between this model and the Jelinski-Moranda Model of Paragraph 4.2.3 is established.

Suppose there is an initial number of N errors present in the program. Suppose n errors have been corrected after τ amount of testing (based upon CPU

time) has elapsed. Then from assumption (e), the hazard rate function at time τ is of the form:

$$Z(\tau) = fK(N - n) \ , \tag{4.449}$$

where f is taken as the linear execution frequency (average instruction rate divided by the number of instructions in the program) and K is an error exposure ratio which relates error exposure frequency to linear execution frequency. The error exposure ratio attempts to account for the fact that code is not executed in a sequential manner, due to numerous loops and branches, and for the variation of the machine state. The variation of the machine state may cause an error associated with a particular instruction to be undetected on a given execution of the instruction.

From assumption (f),

$$\frac{dn}{d\tau} = BZ(\tau) \ , \tag{4.450}$$

where B is the proportionality constant. B is called the error reduction factor. It is the average ratio of the rate of reduction of errors to the rate of failure occurrence. Usually B is positive and less than 1 although there is the situation in which the finding of the error that led to the failure of the program leads to the discovery of additional errors as well. This creates a B larger than 1.

Musa generalizes this relationship by considering,

$$\frac{dn}{d\tau} = BCZ(\tau) \tag{4.451}$$

where B is as before and C is a constant called the testing compression factor. It is the average ratio of rate of detection of errors during testing to that during use. It attempts to account for the greater stress that is placed on a program to uncover program errors during the testing phase in contrast to the operational phase. Usually C is larger than 1 because of this fact.

Now suppose m represents the number of failures experienced in the process of correcting n errors and suppose M is the required number of failures that one needs to experience to uncover all N errors within the program. Then

$$n = Bm \tag{4.452}$$

and

$$N = BM \ . \tag{4.453}$$

The previous equations can be combined to obtain:

$$\frac{dn}{d\tau} = BCZ(\tau) \tag{4.454}$$

$$= BC[fK(N - n)] \tag{4.455}$$

$$= BCfKN - BCfKn \ , \tag{4.456}$$

i.e.,

$$\frac{dn}{d\tau} + BCfKn = BCfKN \tag{4.457}$$

or in terms of the m's,

$$B \frac{dm}{d\tau} + B^2CfKm = B^2CfKM \tag{4.458}$$

or

$$\frac{dm}{d\tau} + BCfKm = BCfKM \ . \tag{4.459}$$

Since $n = m = 0$ at $\tau = 0$, equation (4.457) has the solution

$$n = N \ [1 - \exp(-BCfK\tau)] \ . \tag{4.460}$$

and equation (4.459) has the solution

$$m = M \ [1 - \exp(-BCfK\tau)] \ . \tag{4.461}$$

Since the MTBF is given by:

$$MTBF = \frac{1}{Z(\tau)} \ , \tag{4.462}$$

it can be reexpressed as:

$$MTBF = \frac{1}{Z(\tau)} \tag{4.463}$$

$$= \frac{1}{fK(N - n)} \tag{4.464}$$

$$= \frac{1}{fK(N - N + N \exp(-BCfK\tau))} \ , \text{ using equation (4.460),} \tag{4.465}$$

$$= \frac{1}{fKN \exp(-BCfK\tau)} \ . \tag{4.466}$$

If T_0 is the initial MTBF when testing just begins, i.e., $\tau = 0$, then

$$T_0 = \text{Initial MTBF} = \frac{1}{Z(0)} \tag{4.467}$$

$$= \frac{1}{fKN} \ . \tag{4.468}$$

Thus,

$$MTBF = T_0 \exp (BC\tau/NT_0) \tag{4.469}$$

for any testing time τ. As $\tau \to \infty$, the MTBF $\to \infty$ indicating the improvement in the software as testing proceeds.

The reliability of the program at any future time τ_1 given testing of length τ is found from:

$$R(\tau_1) = \exp \left[- \int_0^{\tau_1} Z(\tau)dx \right] = \exp \left[-\tau_1 Z(\tau)\right]$$
$$= \exp \left[\frac{-\tau_1}{MTBF}\right]. \tag{4.470}$$

Again it can be seen as $\tau \to \infty$ causing both τ_1 and MTBF $\to \infty$, $R(\tau_1) \to 1$ is obtained.

From this basic model, Musa establishes some other useful results (Reference 52). The number of failures Δm that must be detected and corrected to achieve an increase in MTBF from T_1 to T_2 can be shown to be:

$$\Delta m = MT_0 \left[\frac{1}{T_1} - \frac{1}{T_2}\right] . \tag{4.471}$$

The additional execution time required to achieve the increase is:

$$\Delta\tau = \frac{MT_0}{C} \ell n \left(\frac{T_2}{T_1}\right) . \tag{4.472}$$

For the implementation of this model for a reliability analysis, an idea of what the values of these various parameters are is needed. Musa suggests that initial estimates can be obtained from other projects of a similar nature. For some of the parameters reestimation can then be made as the testing progresses. The error reduction factor, B, can be determined by taking data on the number of errors generated while fixing other errors. This information could be obtained from the development of similar programs. Musa reports that B is relatively stable for the programs he considers; it is in the range .94 to 1.00.

The testing compression factor C must also be obtained in a similar manner. If there is no basis for estimation of C, a conservative approach of taking C = 1 is advised. An initial value of M can be obtained from the relationship M = N/B with N being estimated from an idea of the average error rate for programs of a similar nature. Musa notes,[52] from a number of other studies being observed, error rates in the range of 3.36 to 7.98 errors per thousand lines of instruction with a weighted mean of 5.43 errors per thousand lines of instruction. In a later report (Reference 55), Musa employs an estimate of 6.25 errors per thousand lines of instruction. The accuracy of the initial estiamtes for N and hence M do not have to be very high since as the testing progresses, they are reestimated. The

parameter K, the error exposure ratio, must also be estimated initially from programs of a similar nature; however, like M and N, it can be reestimated as the testing progresses. Therefore, the initial accuracy for this estimate can be low as well. Musa observes an average value of 1.31×10^{-6} for K for the various validating projects he considered.

For the reestimation of K and M, suppose $X_1,\ldots,X_m$ are the times between error occurrences. Using assumption (d), Musa[52] establishes the MLEs for M and T_0, the initial MTBF, as the solutions to the following system of equations:

Estimates

$$\hat{T}_0 = C\,\bar{X}\left[1 - \frac{m}{M}\,\hat{\phi}\right] \tag{4.473}$$

and

$$\frac{\hat{M}}{m} - \frac{1}{\sum_{j=\hat{M}-m+1}^{\hat{M}} \frac{1}{j}} = \hat{\phi} \tag{4.474}$$

where

$$\hat{\phi} = \frac{1}{m^2\,\bar{X}} \sum_{i=1}^{m} (i-1)X_i \tag{4.475}$$

and

$$\bar{X} = \frac{\sum_{i=1}^{m} X_i}{m} . \tag{4.476}$$

The estimate of K is then obtained from the relationship:

$$T_0 = \frac{1}{fKN} = \frac{1}{fKBM} ; \tag{4.477}$$

i.e.,

$$\hat{K} = \frac{1}{f\hat{B}\hat{T}_0 M} . \tag{4.478}$$

If τ amount of testing has been completed (measured in CPU), then from the results that have been established earlier:

$$\hat{MTBF} = \hat{T}_0 \exp\left(C\tau/\hat{M}\hat{T}_0\right) \tag{4.479}$$

and the estimated reliability for time τ_1 is:

$$\hat{R}(\tau_1) \simeq \exp(-\tau_1/\hat{MTBF}) \quad . \tag{4.480}$$

Approximate variances for the estimates are also given by Musa[52] as:

$$\text{var}\{\hat{T}_0\} \simeq \frac{\hat{T}_0^2}{m} \tag{4.481}$$

and

$$\text{var}\{\hat{\phi}\} \simeq -\frac{1}{\left(\sum_{j=M+1-m}^{\hat{M}} \frac{1}{j}\right)^2} \left[\frac{-\sum_{j=M+1-m}^{\hat{M}} \frac{1}{j^2}}{\sum_{j=M+1-m}^{\hat{M}} \frac{1}{j}} + \frac{1}{m} \right] \quad . \tag{4.482}$$

Large sample 100X(1-α) percent confidence intervals are then constructed as:

Confidence Intervals

$$\left(\hat{T}_0 - Z_{1-\frac{\alpha}{2}} \sqrt{\text{var}\{\hat{T}_0\}} \; , \; \hat{T}_0 + Z_{1-\frac{\alpha}{2}} \sqrt{\text{var}\{\hat{T}_0\}} \right) \tag{4.483}$$

and

$$\left(\hat{M}_{LOW}, \hat{M}_{UPPER} \right) \tag{4.484}$$

where $\hat{M}_{LOW}$ is chosen to satisfy:

$$\frac{1}{m^2 \bar{X}} \sum_{i=1}^{m} (i-1)X_i = \frac{M_{LOW}}{m} - \frac{1}{\sum_{j=M_{LOW}-m+1}^{M_{LOW}} \frac{1}{j}} + k\sqrt{\text{var}\{\hat{\phi}\}} \tag{4.485}$$

and M_{UPPER} is chosen to satisfy:

$$\frac{1}{m^2 \bar{X}} \sum_{i=1}^{m} (i-1)X_i = \frac{M_{UPPER}}{m} - \frac{1}{\sum_{j=M_{UPPER}-m+1}^{M_{UPPER}} \frac{1}{j}} - k\sqrt{\text{var}\{\hat{\phi}\}} \tag{4.486}$$

with

$$k = \frac{1}{\sqrt{\alpha}} \tag{4.487}$$

and $Z_{1-\frac{\alpha}{2}}$ is taken from a normal table such that:

$$P\left\{Z \geq Z_{1-\frac{\alpha}{2}}\right\} = \frac{\alpha}{2} . \tag{4.488}$$

Musa's Model can be related to Jelinski-Moranda's Model (therefore, to all models which have been shown equivalent to it) by letting,

$$n = m \tag{4.489}$$

and

$$N = M , \tag{4.490}$$

$$\text{which together gives } B = 1, \tag{4.491}$$

$$\text{and } fK = \phi . \tag{4.492}$$

If a point τ is chosen between the occurrence of the $(i - 1)$st error and the ith error, the hazard rate function for Musa's Model is

$$Z(\tau) = fK(N - (i - 1)) \tag{4.493}$$

$$= \phi\,(N - (i - 1)) . \tag{4.494}$$

This is precisely the hazard rate for the Jelinski-Moranda Model.

Also note that:

$$T_0 = \text{Initial MTBF} = \frac{1}{fKN} \tag{4.495}$$

becomes:

$$T_0 = \frac{1}{\phi N} ; \tag{4.496}$$

the correct expression under the Jelinski-Moranda Model. The MTBF, after the discovery of $(i - 1)$ errors for Musa's Model, was shown to be:

$$\text{MTBF} = \frac{1}{Z(\tau)} = \frac{1}{fK(N - i + 1)}$$

for

$$t_{i-1} \leq \tau \leq t . \tag{4.497}$$

With the previous relationships, then:

$$\text{MTBF} = \frac{1}{fK(N - i + 1)} \tag{4.498}$$

$$= \frac{1}{\phi(N - i + 1)} \quad , \tag{4.499}$$

again the correct expression for the MTBF under the Jelinski-Moranda Model.

The importance of Musa's Model is in its development of resource allocation and the relationship between CPU time and wall clock time. The resources (failure identification personnel, failure correction personnel, and computer time) influence the failure detection rate during the testing process. At any point in the testing cycle, one of these resources limits the other two and thus, the error detection rate. For example, if the number of failure correction personnel is insufficient to handle the errors detected by the failure identification personnel, a backlog of errors develops, slowing down the testing process. Usually the testing process involves from one to three periods, each one characterized by a different limiting resource. At the start of testing, when numerous errors are discovered, the limiting factor is the failure correction personnel. As the testing progresses and longer intervals between failures are observed, the failure correction personnel utilization drops, while the failure identification personnel becomes the limiting factor. Finally, at longer failure intervals, the use of the computer becomes the prime limiting factor. Musa's Model attempts to utilize the knowledge of these limiting resources to relate execution time with the passage of calendar time. Suppose $\frac{dt_I}{d\tau}$, $\frac{dt_F}{d\tau}$, and $\frac{dt_C}{d\tau}$ are the instantaneous calendar time to execution time ratios that result from the effects of each of the resource constraints taken alone. The index I denotes failure identification personnel, F denotes failure correction personnel, and C denotes computer use. An increment in calendar time, Δt, is taken to be proportional to the average amount by which the limiting resource constraints testing over a given execution time segment; that is,

$$\Delta t = \int_{\tau_1}^{\tau_2} \max\left(\frac{dt_I}{d\tau} , \frac{dt_F}{d\tau} , \frac{dt_C}{d\tau} \right) d\tau \ . \tag{4.500}$$

From assumption (h), the resource requirements associated with a change in MTBF from T_1 to T_2 can be approximated by:

$$\Delta\chi_k \simeq \theta_k \, \Delta\tau + \mu_k \, \Delta m \tag{4.501}$$

where $\Delta\tau$ is the increment of execution time, Δm is the increment of failures experienced, θ_k is an execution time coefficient of resource expenditure, and μ_k is a failure coefficient of resource expenditure for k = I, F, and C.

Suppose P_k represents the number of available personnel, k = I, F, or the available number of computer shifts, k = C. Suppose ρ_k denotes the utilization

factor for the kth resource, [from assumption (i), $\rho_I = 1$]. Then the effective available amount of the kth resource is $\rho_k P_k$. From this basic formulation, Musa[52] derives the following correspondence between the resources and the calendar time:

$$\Delta\tau = \frac{MT_0}{C} \int_{T_1}^{T_2} \frac{1}{T} \max \left[\frac{\theta_k T + C\mu_k}{P_K \rho_K T} \right] dT \tag{4.502}$$

$$= MT_0 \sum_k \frac{1}{P_k \rho_k} \left[\frac{\theta_k}{C} \ln \left(\frac{T_{k_2}}{T_{k_1}} \right) + \mu_k \left(\frac{1}{T_{k_1}} - \frac{1}{T_{k_2}} \right) \right] , \tag{4.503}$$

where the index k can have the values C, F, or I, and the quantities T_{k_1} and T_{k_2} represent the MTBF at the boundary of these periods. These boundaries are the values T_1, T_2, and the transition points

$$T_{kk'} = \frac{C(P_k \mu_k \rho_k - P_{k'} \mu_k \rho_{k'})}{P_{k'} \rho_{k'} \theta_k - P_k \rho_k \theta_{k'}} \tag{4.504}$$

for k, k′ = I, F, C. The transition points are those values of T at which the derivative of calendar time, with respect to execution time for one resource, becomes greater than another. The resource k that is limiting for any given MTBF, T, is the one that maximizes:

$$\frac{\theta_k T + C\mu_k}{P_k \rho_k T} . \tag{4.505}$$

From assumption (j), it can be established that the utilization factor for failure correction personnel is of the form:

$$\rho_F = (1 - P^{1/P_F})\ 1/Q \tag{4.506}$$

where Q is the established limitation of error queue length (at a specified probability P) for any debugger.

As can be seen from the formulation of the model, the data required for implementation of the complete model can be quite extensive.

Data Requirements

[Execution Part]

(a) The linear execution frequency, f.

(b) An initial estimate of the error exposure ratio, K. (The accuracy of the initial estimate can be low).

(c) The error reduction factor, B.

(d) The testing compression factor, C.

(e) An initial estimate of the total number of errors, N. (The accuracy of the estimate can also be low since it is reestimated during testing.)

(f) The times (measured in CPUs) between error occurrences, τ_i's.

Execution/Calendar Time Part

(g) The available resources for both testing and correction personnel and the number of computer shifts; i.e., P_I, P_F, and P_C.

(h) The utilization factor for each of these resources, i.e., ρ_I (= 1), ρ_F, and ρ_C.

(i) The execution time coefficent of resource expenditure for each resource; i.e., θ_I, θ_F (= 0 usually) and θ_C.

(j) The failure coefficient of resource expenditure for each resource; i.e., μ_I, μ_F, and μ_C.

(k) The maximum error queue length, Q, for a debugger.

(l) The probability, P, that the error queue length is no larger than Q.

Two extensions to Musa's Model are briefly discussed here. The first appears in a paper by Chenoweth (Reference 57). In that paper, the error reduction factor, B, is generalized to the form $B_0 e^{a\tau}$ where B_0 is the initial error reduction factor and a is the exponential slope of execution time. Chenoweth argues that for a certain class of software programs, B, appears to be exponentially increasing. The basis of the increase is probably due to a programmer learning curve phenomena. The parameter a can be estimated from the relationship:

$$\sum_{i=1}^{j} n(\tau_i) = B_0 \sum_{i=1}^{j} ie^{a\tau_i} \qquad (4.507)$$

where $n(\tau_i)$ is the number of errors corrected by time τ_i for a specified j (j=1, ...,m the number of errors observed) and B_0 is obtained from a project of a similar nature or using this relationship.

The second modification is contained in a paper by Musa and Iannino (Reference 58). The modification can actually be applied to many of the previously considered models, but it is illustrated in the report on the execution time

theory model. The paper describes a method of adjusting the lengths of the intervals between software failures to compensate for programs that are undergoing variations in length due to integration or design changes. The models considered so far have been applied to essentially complete programs. In the testing process, all of the code is being executed at one time or another. Frequently, however, only part of a program is tested and other parts are added as testing proceeds. By ignoring these variations, estimated MTBFs in the early stages of a project tend to be optimistic. The method presented in this paper attempts to account for the variations by adjusting the observed failure intervals to values that would have been for a program in its final configuration with complete inspection. The adjusted values are used in the various models in the exact manner as if they had been the actual data. The reader is referred to Musa and Iannino's paper for details.

4.2.12 Brooks and Motley's Models

The last models discussed in this section are the Binomial Model and the Poisson Model formulated by Brooks and Motley of the IBM Corporation (Reference 59). Their models try to account for the fact that in a given testing period not all of the program is tested equally, and in the development of a program, only some portion or modules may be available for testing. In addition, in the correction of discovered errors, additional errors may be introduced. Each of the models make the following assumptions:

Model Assumptions

(a) The number of software errors detected on each test occasion is proportional to the number of errors at risk for detection which is, in turn, proportional to the remaining number of errors.

(b) The proportionality factor or probability (denoted as q for the binomial model, and ϕ for the Poisson) of detecting any error during a specified unit interval of testing is constant over all occasions and independent of error detections.

(c) The errors reintroduced in the correction process are proportional to the number of errors detected.

For their formulation, Brooks and Motley, develop the models both for a module application, in case only module testing is done, and for the entire program system testing.

4.2.12.1 Binomial Model (module). Suppose a module, the jth, from the program is given for testing for the first time. Then the expected number of error occurrences in that module in the first unit interval of the test occasion is:

$$n_{1j} = w_j N q. \tag{4.508}$$

This is obtained from assumptions (a) and (b), where w_j is the weight assigned to module j, N is the total number of errors in the system at the beginning of testing, and q is the error detection probability given in assumption (b). Brooks and Motley define a test occasion as:

> "an event of error data collection; each occasion should have a time interval associated with it; otherwise, the implication to the model is that all test occasions are of equal length of time.... One additional important assumption made here is that one occasion be comparable to every other occasion in terms of the time spent (testing effort) in detecting errors."

The weight factor can be taken as the ratio of the size of the module (as measured by number of lines of source code or object program size) to the total program size.

For the second unit interval of testing on the jth module, the expected number of errors to be detected is:

$$[w_jN - w_jNq]q = [w_jN(1 - q)]q. \tag{4.509}$$

There were w_jNq expected errors in the first unit interval of time leaving $w_jN - w_jNq$ errors subject to detection in the second. Thus the expected number of errors in the second time interval is: [number of errors subject to detection] $\cdot q = [w_jN(1 - q)]q$. In general, for the ith unit interval of time in the first testing session, the expected number of errors is:

$$w_jN(1 - q)^{i-1}q. \tag{4.510}$$

The total number of errors expected for the entire first testing occasion is then:

$$n_{1j} = \sum_{i=1}^{K_{1j}} w_jN(1 - q)^{i-1}q \tag{4.511}$$

$$= w_jN\left[1 - (1 - q)^{K_{1j}}\right] \tag{4.512}$$

$$= w_jNq_{1j} \tag{4.513}$$

where K_{1j} is the number of unit test intervals making up the first test occasion, or the total test effort expended on module j during the first test occasion, and $q_{1j} = [1 - (1 - q)^{K_{1j}}]$.

When module j is tested for the second test occasion, the number of errors at risk in module j is:

$$w_j N - n_{1j} + rn_{1j} \tag{4.514}$$

where n_{1j} is the number of errors detected in the first testing period and rn_{1j} is the number introduced into the program as a result of correcting those n_{1j} errors, (assumption (c)). The total expected number of errors in the second test period can be shown as was done for the first to be:

$$n_{2j} = (w_j N - \alpha n_{1j}) q_{2j}$$

where $\alpha = 1 - r$, (the probability of correcting code without introducing new errors) and

$$q_{2j} = (1 - (1 - q)^{K_{2j}}) . \tag{4.515}$$

In general for the ith testing period, the expected number of errors detected is:

$$n_{ij} = (w_j N - \alpha N_{i-1,j}) q_{ij} \tag{4.516}$$

$$= \tilde{N}_{ij} q_{ij} \tag{4.517}$$

where

$$N_{i-1,j} = \sum_{m=1}^{i-1} n_{mj}, \tag{4.518}$$

(the total number of errors found up to the ith testing period),

$$q_{ij} = (1 - (1 - q)^{K_{ij}}) , \tag{4.519}$$

and

$$\tilde{N}_{ij} = (w_j N - \alpha N_{i-1,j}) , \tag{4.520}$$

the number of errors remaining in the jth module. One notices that K_{ij}, the amount of testing effort expended in the ith period, can be different from one testing period to the next. The only restriction is that the probability, q, of detection for any error is the same from period to period. This means that the times can vary for each testing period, but the testing approach should be the same.

Brooks and Motley establish the MLEs of the three unknowns, (N, q, α) of their model as the solutions of the following equations.

Estimates - Maximum Likelihood

$$\frac{\partial \ell nL}{\partial N} = 0 = \sum_{i=1}^{K} \sum_{j=1}^{J} \left[w_j \ell n \left(\frac{\bar{N}_{ij}}{\bar{N}_{ij} - n_{ij}} \right) + w_j K_{ij} \ell n(1 - q) \right] \tag{4.521}$$

$$\frac{\partial \ell nL}{\partial q} = 0 = \sum_{i=1}^{K} \sum_{j=1}^{J} \left[\frac{n_{ij} K_{ij}}{\left(1 - (1 - q)^{K_{ij}}\right)} - K_{ij} \bar{N}_{ij} \right] \tag{4.522}$$

and

$$\frac{\partial \ell nL}{\partial \alpha} = 0 = \sum_{i=1}^{K} \sum_{j=1}^{J} \left[N_{i-1,j} \ell n \left(\frac{\bar{N}_{ij}}{\bar{N}_{ij} - n_{ij}} \right) \right] + N_{i-1,j} K_{ij} \ell n(1 - q) \tag{4.523}$$

where the likelihood function

$$L = \prod_{i=1}^{K} \prod_{j=1}^{J} \binom{\bar{N}_{ij}}{n_{ij}} q_{ij}^{n_{ij}} (1 - q)^{\bar{N}_{ij} - n_{ij}}, \tag{4.524}$$

K = the number of test occasions, J = number of modules in the system, and n_{ij} is the actual number of errors observed on the ith testing occasion of the jth module. These equations do not have a solution if $\bar{N}_{ij} - n_{ij}$ becomes negative. It could happen that the effective number of errors at risk, $\bar{N}_{ij}$, becomes smaller than the actual number of errors observed for the jth module on the ith testing occasion. In that situation, it is recommended that the system model be applied.

4.2.12.2 Poisson (module). As in the Binomial Model, suppose $\bar{N}_{ij} = (w_j N_j - \alpha N_{i-1,j})$ is the effective number of errors at risk in module j at the beginning of the ith testing period. Using assumption (b), the expected error detection rate for the first unit interval of length t is $\bar{N}_{ij}\phi$. Thus the expected number of errors that are detected is the error detection rate, $\bar{N}_{ij}\phi$, times the length of the testing interval; t, i.e.,

$$n_{ij} = \bar{N}_{ij}\phi t. \tag{4.525}$$

At the end of the first unit time interval testing period of module j, the number of errors remaining is:

$$\bar{N}_{ij} - \bar{N}_{ij}\phi t = \text{number of errors in the module at the beginning of the first period minus the number of errors detected during the first testing period.} \tag{4.526}$$

The error detection rate for the second testing period is:

$$[\bar{N}_{ij} - \bar{N}_{ij}\phi t]\phi = [\bar{N}_{ij}(1 - \phi t)]\phi \tag{4.527}$$

so the expected number of errors is therefore:

$$[\bar{N}_{ij}(1 - \phi t)]\phi t$$

for the second unit interval of testing. In general, for the t_{ij} unit interval of testing of the jth module in the ith testing period:

(a) The number of errors remaining at the beginning of the interval

$$= \bar{N}_{ij}(1 - \phi t)^{t_{ij}-1} \tag{4.528}$$

(b) The expected error detection rate is:

$$= \bar{N}_{ij}(1 - \phi t)^{t_{ij}-1}\ \phi \tag{4.529}$$

and thus,

(c) The expected number of errors detected is:

$$\bar{N}_{ij}(1 - \phi t)^{t_{ij}-1}\ \phi t\ . \tag{4.530}$$

The length of the unit testing interval, t, is then normalized to 1 (example: 1 day, 1 week, etc). Thus the total number of expected errors for the jth module on the ith testing occasion is:

$$n_{ij} = \sum_{\ell=1}^{t_{ij}} \bar{N}_{ij}\phi(1 - \phi)^{\ell-1} \tag{4.531}$$

$$= \bar{N}_{ij}\phi_{ij} \tag{4.532}$$

where

$$\phi_{ij} = 1 - (1 - \phi)^{t_{ij}}\ . \tag{4.533}$$

The likelihood function is then:

$$L = \prod_{i=1}^{K} \prod_{j=1}^{J} \frac{(\bar{N}_{ij}\phi_{ij})^{n_{ij}}\ e^{-\bar{N}_{ij}\phi_{ij}}}{n_{ij}!}\ . \tag{4.534}$$

The MLEs are then obtained as the solution to the equations:

Estimates - Maximum Likelihood

$$\frac{\partial \ell nL}{\partial N} = 0 = \sum_{i=1}^{K} \sum_{j=1}^{J} w_j \left[\frac{n_{ij}}{\tilde{N}_{ij}} - \phi_{ij} \right] \tag{4.535}$$

$$\frac{\partial \ell nL}{\partial \phi} = 0 = \sum_{i=1}^{K} \sum_{j=1}^{J} t_{ij}(1 - \phi)^{t_{ij}} \left[\frac{n_{ij}}{1 - (1 - \phi)^{t_{ij}}} - \tilde{N}_{ij} \right] \tag{4.536}$$

and

$$\frac{\partial \ell nL}{\partial \alpha} = 0 = \sum_{i=1}^{K} \sum_{j=1}^{J} N_{i-1,j} \left[\frac{n_{ij}}{\tilde{N}_{ij}} - \phi_{ij} \right] \tag{4.537}$$

4.2.12.3 Binomial (system). For this model, the overall program is considered as a whole. Keeping the same notation as was used in the Binomial (module) paragraph, if J_i is the index set of those modules tested on occasion i, then the total number of errors remaining in the program and subject to detection is

$$\tilde{N}_i = \sum_{j \varepsilon J_i} (w_j N - \alpha N_{i-1,j}) \ . \tag{4.538}$$

Since the system is being considered as a whole, the test effort involved for the system can be considered as a whole rather than on a modular basis. The

$$q_{ij} = [1 - (1 - q)^{K_{ij}}] \tag{4.539}$$

of the modular section is then replaced by:

$$q_i = [1 - (1 - q)^{K_i}] \tag{4.540}$$

where K_i is the system test effort (e.g., computer CPU time) expended on the ith test occasion. Combining the previous information, the total number of expected errors in the system for the ith testing occasion is:

$$n_i = \tilde{N}_i q_i \ . \tag{4.541}$$

The likelihood equation for the system is therefore:

$$L = \prod_{i=1}^{K} \binom{\tilde{N}_i}{n_i} q_i^{n_i} (1 - q_i)^{\tilde{N}_i - n_i} \tag{4.542}$$

where

$$n_i = \sum_{j=1}^{J} n_{ij} = \text{total number of errors found in the program on the ith testing occasion.} \quad (4.543)$$

The MLEs are then obtained as the solutions to the following set of equations:

Estimates - Maximum Likelihood

$$\frac{\partial \ell nL}{\partial N} = 0 = \sum_{i=1}^{K} \left(\ell n \left[\frac{\bar{N}_i}{\bar{N}_i - n_i} \right] + K_i \ell n(1 - q) \right) \sum_{j \varepsilon J_i} w_j \quad (4.544)$$

$$\frac{\partial \ell nL}{\partial q} = 0 = \sum_{i=1}^{K} \left[\frac{n_i K_i}{1 - (1 - q)^{K_i}} - K_i \bar{N}_i \right] \quad (4.545)$$

$$\frac{\partial \ell nL}{\partial \alpha} = 0 = \sum_{i=1}^{K} \left[\ell n \left(\frac{\bar{N}_i}{\bar{N}_i - n_i} \right) + K_i \ell n(1 - q) \right] \sum_{j \varepsilon J_i} N_{i-1,j} \, . \quad (4.546)$$

Again if $\bar{N}_i - n_i$, the difference between the expected number of remaining errors and the actual number of errors found on the ith testing occasion is negative, no solution exists to these equations.

4.2.12.4 Poisson (system). Using the expression for the number of errors at risk at the start of the ith testing occasion for the system, i.e.,

$$\bar{N}_i = \sum_{j \varepsilon J_i} (w_j N - \alpha N_{i-1,j}) \, , \quad (4.547)$$

the total expected number of errors is:

$$n_i = \bar{N}_i \phi_i \quad (4.548)$$

where

$$\phi_i = 1 - (1 - \phi)^{t_i} \quad (4.549)$$

and t_i is the total time spent for the ith testing occasion.

The likelihood function is therefore:

$$L = \prod_{i=1}^{K} \frac{(\bar{N}_i \phi_i)^{n_i} e^{-\bar{N}_i \phi_i}}{n_i!} \tag{4.550}$$

where n_i is the total number of errors detected during the ith testing occasion. The MLEs are again obtained as the solutions to the following system of equations:

Estimates - Maximum Likelihood

$$\frac{\partial \ell n L}{\partial N} = 0 = \sum_{i=1}^{K} \left(\sum_{j \varepsilon J_i} w_j \right) \left(\frac{n_i}{\bar{N}_i} - \phi_i \right) \tag{4.551}$$

$$\frac{\partial \ell n}{\partial q} = 0 = \sum_{i=1}^{K} t_i (1 - \phi)^{t_i} \left[\frac{n_i}{1 - (1 - \phi)^{t_i}} - \bar{N}_i \right] \tag{4.553}$$

and

$$\frac{\partial \ell n L}{\partial \alpha} = 0 = \sum_{i=1}^{K} \left(\sum_{j \varepsilon J_i} N_{i-1,j} \right) \left(\frac{n_i}{\bar{N}_i} - \phi_i \right) . \tag{4.554}$$

The various sets of equations given in the previous paragraphs can all be solved using the Newton-Raphson method with the warning about the $(\bar{N}_i - n_i)$'s becoming negative applying as in the binominal formulations.

All of the models by Brooks and Motley were applied to real life and simulated data. One criticism that might be made against their models is the assumption of a constant detection probability, q. In a testing environment, the usual situation is that q varies over time. This is due to limiting resources, the easier errors are found at the beginning, while the hidden errors are discovered much later and at greater effort, and there is a learning curve effect on the testers. Brooks and Motley do however try to account for this by considering an extension to their basic models. They allow for the probability of detection to increase at a constant amount until it reaches a point where it levels off. The resulting equations for this extension are very complex and would be difficult to implement on a computer. The reader is referred to their paper for additional details.

4.3 BAYESIAN MODELS

The class of models considered in this paragraph formulated software reliability modeling in a Bayesian framework. The models employ a "subjective"

approach to the meaning of software reliability in constrast to the traditional "frequentist" approach. Previous models only allowed for change in the reliability of a program whenever an error was discovered and subequently corrected. Bayesian models take the subjective viewpoint that as the software is tested, if no errors are discovered, there is more confidence in the program and this is reflected in increasing reliability. The reliability of a program should be a reflection of the number of errors discovered and the length of error-free testing time periods.

Another important argument given in support of a Bayesian approach deals with counting errors. All of the models considered so far assume that the hazard rate function is directly proportional to the number of errors in the program at the time. From this assumption, it is directly determined that the reliability is a function of this count. This is the reason the models considered in the previous paragraph are concerned with estimating this total. The Bayesian approach argues that a program with two or more errors in little exercised portions of code is considered more reliable than one with only one error in a frequently executed section of code. The estimation of the total number of errors present can be of use to the software manager in making determinations of resource allocation, but it should not be the driving factor in reliability considerations. One should be concerned with measuring operational reliability.

A number of models which attempt to do this are now considered.

4.3.1 Littlewood's Bayesian Debugging Model

The first model considered within this class was proposed by Bev Littlewood of the City University of London.[60,61,62,63,64] The model reformulates the Jelinski-Moranda Model (Paragraph 4.2.3) into a Bayesian framework. The Jelinski-Moranda Model postulates that, at any point in time, the error rate is proportional to the number of errors remaining in the program. This is expressed as for any time t, for $t_{i-1} \leq t < t_i$,

$$Z(t) = \phi(N - i + 1) \tag{4.555}$$

where the t_i's are the times of error occurrences. By making the assumption that the times between error occurrences, i.e., $X_i = t_i - t_{i-1}$, follow an exponential distribution, the probability density function for X_i is seen to be:

$$f(X_i) = \phi(N - i + 1)\exp[-\phi(N - i + 1)X_i]. \tag{4.556}$$

The model inherently makes the assumption that all errors contribute equally; namely ϕ, to the overall error rate. The Bayesian viewpoint objects to this assumption.[65] Each error does not contribute equally since the correction of errors in the beginning of the testing phase, has more of an effect on the program than ones corrected later. Again the argument that a program with two errors in rarely exercised code is more reliable than a program with only one error in a

frequently exercised section surfaces. All errors, therefore, do not contribute equally. Littlewood postulates that the error rate

$$\lambda_i = Z(t) = \phi(N - i + 1), \quad t_{i-1} \leq t < t_i \tag{4.557}$$

should be treated as a random variable, not as a constant. By assuming that the remaining errors have different occurence rates ϕ_1, ϕ_2, ϕ_{N-i+1}, the overall failure rate is then:

$$\lambda_i = \phi_1 + \phi_2 + \ldots + \phi_{N-i+1} \quad . \tag{4.558}$$

By treating the ϕ_i's as random variables (since it is not known what they are), the overall rate as a random variable is obtained. (Notice that if all of the ϕ_i's are assumed to have a degenerate distribution at the point ϕ, i.e., $\phi_1=\phi_2 = \ldots = \phi_{N-i+1} = \phi$ with probability 1, then $\lambda_i = \phi(N - i + 1)$). The specific assumptions for this model are:

Model Assumptions

(a) The individual failure rates of the errors in the program are assumed to be independent random variables each with a prior distribution that is assumed gamma with parameters α and β, i.e.,

$$g(\phi_i) = g(\phi_j) = \frac{\beta^{\alpha}\phi^{\alpha-1}\, e^{-\beta\phi}}{\Gamma(\alpha)} \quad , \ \phi > 0 \tag{4.559}$$

for all i and j.

(b) For a given error rate, λ_i, the time between error occurrence $X_i = t_i - t_{i-1}$ is assumed to be exponential with mean $1/\lambda_i$; i.e.,

$$f(X_i \mid \lambda_i) = \lambda_i e^{-\lambda_i X_i} \qquad X_i > 0 \ . \tag{4.560}$$

(c) $\lambda_i = \phi_1 + \phi_2 + \ldots + \phi_{N-i+1}$ (4.561)

after i - 1 errors have been detected and corrected.

(d) When a software error is detected, it is immediately corrected without the introduction of additional errors.

(e) The software is operated in a similar manner as the anticipated operational usage.

The model is developed as follows. Suppose $X_1, \ldots, X_n$ are the times between errors occurrences, i.e.,

$$X_i = t_i - t_{i-1} \tag{4.562}$$

(Preferably X_i is measured in CPU time rather than wall clock time.) At the time the ith error is discovered and corrected from assumption (c),

$$\lambda_{i+1} = \phi_1 + \phi_2 + \ldots + \phi_{N-i} \quad . \tag{4.563}$$

Now suppose the occurrence rate ϕ_k for any one of the remaining N - i errors is considered. The density function for ϕ_k is pdf(ϕ_k | given that error was not found in (0,t)), where t is the current testing time

$$= \int \frac{P\{\text{no failure by that error in } (0,t) \mid \phi_k = \phi_k\}\ \text{pdf}(\phi_k)}{P\{\text{no failure by that error in } (0,t) \mid \phi_k = \phi_k\}\ \text{pdf}(\phi_k) d\phi_k} \tag{4.564}$$

$$= \frac{e^{-\phi_k t} \cdot \dfrac{\beta^\alpha \phi_k^{\alpha-1} e^{-\beta\phi_k}}{\Gamma(\alpha)}}{\displaystyle\int_0^\infty e^{-\phi_k t} \frac{\beta^\alpha \phi_k^{\alpha-1} e^{-\beta\phi_k}}{\Gamma(\alpha)} d\phi_k} \tag{4.565}$$

$$= \frac{(\beta+t)^\alpha \phi_k^{\alpha-1} e^{-(\beta+t_i)\phi_k}}{\Gamma(\alpha)} \quad . \tag{4.566}$$

Thus ϕ_k has a density function that is also gamma with parameters α and β+t. Since λ_{i+1} is a sum of independent, identically distributed random variables, λ_{i+1} is also gamma with parameters (N - i)α and β + t. Thus, the unconditional distribution of the time to the next failure X_{i+1} is:

$$f(X_{i+1}) = \int_0^\infty f(X_{i+1} \mid \lambda_{i+1})\ g(\lambda_{i+1}) d\lambda_{i+1} \tag{4.567}$$

$$= \int_0^\infty \frac{\lambda_{i+1} e^{-\lambda_{i+1} X_{i+1}} (\beta + t)^{(N-i)\alpha} \lambda_{i+1}^{(N-1)\alpha-1} e^{-(\beta+t)\lambda_{i+1}} d\lambda_{i+1}}{\Gamma[(N - i)\alpha]} \tag{4.568}$$

$$= \int_0^\infty \frac{\lambda_{i+1}^{[(N-1)\alpha+1]-1} e^{-(\beta+t+x_{i+1})\lambda_{i+1}} (\beta+t)^{(N-i)\alpha}}{\Gamma[(N - i)\alpha]}\ d\lambda_{i+1} \tag{5.569}$$

$$= \frac{(\beta + t)^{(N-i)\alpha}}{\Gamma[(N - i)\alpha]} \cdot \frac{\Gamma[(N - i)\alpha + 1]}{(\beta + t + X_{i+1})^{(N-i)\alpha+1}} . \tag{4.570}$$

$$\int_0^\infty \frac{\lambda_{i+1}^{(N - i)\alpha + 1 - 1} (\beta + t + X_i)^{(N - i)\alpha + 1}}{\Gamma((N - i)\alpha + 1)} e^{-(\beta + t + X_i)\lambda_{i+1}} d\lambda_{i+1}$$

$$= \frac{[(N - i)\alpha] (\beta + t)^{(N-i)\alpha}}{(\beta + t + X_{i+1})^{(N-1)\alpha+1}} \qquad X_{i+1} > 0 . \tag{4.571}$$

This is a Pareto distribution. From this basic result, Littlewood derives a number of quantities. The reliability function after i errors are discovered is found as:

$$R(x) = \{X_{i+1} > x\} \tag{4.572}$$

$$= 1 - P\{X_{i+1} \leq x\} \tag{4.573}$$

$$= 1 - \int_0^x f(X_{i+1})dX_{i+1} \tag{4.574}$$

$$= \left[\frac{(\beta + t)}{(\beta + t + x)}\right] (N - i)\alpha \tag{4.575}$$

The failure rate function is then obtained as:

$$Z(x) = -\frac{R'(x)}{R(x)} = \frac{(N - i)\alpha}{(\beta + t + x)} . \tag{4.576}$$

Thus, the failure rate, immediately after i errors have been discovered and t amount of testing has been employed, is:

$$Z(0) = \frac{(N - i)\alpha}{\beta + t} . \tag{4.577}$$

Notice how this unconditional failure rate changes after testing progresses. As t gets larger, the hazard rate decreases reflecting the increased confidence in the program. The hazard rate also decreases whenever an error is discovered and corrected.

The MTBF is found from the Pareto distribution as:

$$MTBF = E\{X_{i+1}\} = \int_0^\infty X_{i+1} f(X_{i+1})dX_{i+1} \tag{4.578}$$

$$= \frac{\beta + t}{(N - i)\alpha - 1} \tag{4.579}$$

which exists as long as $(N - i)\alpha > 1$.

From this basic model formulation, Littlewood approaches the problem of prediction of future reliability in one of two ways. The reliability of the program can be estimated after some specified execution time has elapsed or after some specified number of errors has been removed. The development of these two approaches is provided in Reference 61 and is not repeated here; however, the uses of those results are.

For the first approach, suppose t_i amount of testing is performed and i errors are discovered and corrected. Now suppose Δt additional amount of testing is done. Then the reliability of the program at the time $t_i + \Delta t$ is shown to be:

$$R(x) = \left[1 - \left(\frac{\beta + t_i}{\beta + t_i + \Delta t} \right)^{\alpha} + \left(\frac{\beta + t_i}{\beta + t_i + \Delta t + x} \right)^{\alpha} \right]^{N-i} . \tag{4.580}$$

From this relationship, the amount of additional testing needed in order to achieve a target reliability can be determined. If the desired reliability is r for a specified error-free run time of x_0, then the additional testing time required is the value Δt that solves the equation:

$$r = \left[1 - \left(\frac{\beta + t_i}{\beta + t_i + \Delta t} \right)^{\alpha} + \left(\frac{\beta + t_i}{\beta + t_i \Delta t + x_0} \right)^{\alpha} \right]^{N-i} . \tag{4.581}$$

Littlewood also shows for this approach that the required additional testing time Δt to achieve a specified target failure rate, λ_0, is:

$$\Delta t = \left[\frac{(N - i)\alpha(\beta + t_i)^{\alpha}}{\lambda_0} \right]^{1/(\alpha+1)} - (\beta + t_i) . \tag{4.582}$$

For the second approach, suppose i errors are observed and corrected. Interest lies in the times between error occurrence of the next k errors, i.e., x_{i+j}, $j=1,\ldots,k$. Littlewood first derives the distribution of:

$$Z = \left[\frac{\Lambda(\beta + t_i)}{(N - i - k)\alpha} \right] \tag{4.583}$$

where Λ is the failure rate at the occurrence of the i + k error. From the previous results, the following is obtained:

$$\Lambda = \frac{(N - i - k)\alpha}{\beta + t_i + \sum_{j=i+1}^{i+k} X_i} . \tag{4.584}$$

The distribution of Z is beta with parameters N-i-k+1 and k, so that the expected value of Λ can then be obtained as:

$$E\{\Lambda\} = \frac{(N - i - k)\alpha}{\beta + t_i} \cdot \frac{(N - i - k + 1)\alpha}{(N - i - k + 1)\alpha + 1} \cdot \cdot \cdot \frac{(N - i)}{(N - i)\alpha + 1} \cdot \tag{4.585}$$

From this result, the number of additional error corrections, k_0, that are required to make the $E\{\Lambda\}$ less than a desired level λ_0 can be established. This is the smallest integer k_0 satisfying:

$$\frac{(N - i - k_0)\alpha}{\beta + t_i} \cdot \frac{(N - i - k + 1)\alpha}{(N - i - k_0 + 1)\alpha + 1} \cdot \cdot \cdot \frac{(N - i)}{(N - i)\alpha + 1} < \lambda_0 \cdot \tag{4.586}$$

It might also be asked how many additional error corrections are necessary in order to be at least γ percent certain that $\Lambda < \lambda_0$. This is the smallest integer k_0 such that:

$$P\left\{ Z < \left[\frac{(\beta + t_i)\ \lambda_0}{(N - i - k_0)\alpha} \right]^{\alpha} \right\} > \gamma \tag{4.587}$$

where Z is from a beta distribution with parameters N-i+k_0+1 and k_0.

For this model, there are three unknowns: N, α, and β. They can be estimated using the maximum likelihood procedure or least squares. If X_i, i = 1,...,n is the time between error occurrences, then from the assumptions, the likelihood function is:

$$L(N,\alpha,\beta) = \prod_{i=1}^{n} f(X_i\ x_{i-1},\ldots,x_1) \tag{4.588}$$

$$= \frac{\prod_{i=1}^{n} (N - i + 1)\alpha(\beta + t_{i-1})^{(N - i + 1)\alpha}}{(\beta + t_{i-1} + X_i)^{(N - i + 1)\alpha + 1}} \tag{4.589}$$

where

$$t_{i-1} = \sum_{j=1}^{i-1} x_j \tag{4.590}$$

is the time of occurrence of the (i - 1)st error.

The MLEs $\hat{N}_L$, $\hat{\alpha}_L$, and $\hat{\beta}_L$ are the estimates which maximize:

Estimates - Maximum Likelihood

$$L(\hat{N}_L, \hat{\alpha}_L, \hat{\beta}_L) = \max_{N,\alpha,\beta} L(N,\alpha,\beta) \quad . \tag{4.591}$$

Littlewood points out that this maximization search can be restricted to the two-dimensional space of N and β as:

$$\hat{\alpha}_L = \frac{n}{\sum_{i=1}^{n-1} \ell n\left[\frac{\hat{\beta}_L + t_i}{\hat{\beta}_L + t_n}\right] + \hat{N}_L \ell n\left[\frac{\hat{\beta}_L + t_n}{\hat{\beta}_L}\right]} \quad . \tag{4.592}$$

The least squares estimates are those N, α, and β which minimize

$$S(N, \alpha, \beta) = \sum_{i=1}^{n} \left[X_i - \left(\frac{\beta + t_{i-1}}{(N - i + 1)\alpha - 1}\right)\right]^2 \tag{4.593}$$

using equation (4.579). The least squares estimates $\hat{N}_{L,LS}$, $\hat{\beta}_{L,LS}$, and $\hat{\alpha}_{L,LS}$ are chosen so that:

$$S(\hat{N}_{L,LS}, \hat{\beta}_{L,LS}, \hat{\alpha}_{L,LS}) = \min_{(N,\alpha,\beta)} S(N, \alpha, \beta) \tag{4.594}$$

and are found as the solution to the following system of equations:

$$\frac{\partial S}{\partial N} = 0, \quad \frac{\partial S}{\partial \alpha} = 0, \text{ and } \frac{\partial S}{\partial \beta} = 0. \tag{4.595}$$

The least squares estimates are then the solutions to the equations:

Estimates -Least Squares

$$\sum_{i=1}^{n} \frac{X_i}{(\hat{N}_{L,LS} - i + 1)\hat{\alpha}_{L,LS} - 1} = \sum_{i=1}^{n} \frac{\hat{\beta}_{L,LS} + t_{i-1}}{[(\hat{N}_{L,LS} - i + 1)\hat{\alpha}_{L,LS} - 1]^2} \tag{4.596}$$

$$\sum_{i=1}^{n} \frac{X_i(\hat{\beta}_{L,LS} + t_{i-1})}{[(\hat{N}_{L,LS} - i + 1)\hat{\alpha}_{L,LS} - 1]^2} = \sum_{i=1}^{n} \frac{(\hat{\beta}_{L,LS} + t_{i-1})^2}{[(\hat{N}_{L,LS} - i + 1)\hat{\alpha}_{L,LS} - 1]^3} \tag{4.597}$$

and

$$\sum_{i=1}^{n} \frac{X_i(\hat{N}_{L,LS} - i + 1)\,(\hat{\beta}_{L,LS} + t_{i-1})}{[(\hat{N}_{L,LS} - i + 1)\,\hat{\alpha}_{L,LS} - 1]^2} = \sum_{i=1}^{n} \frac{(\hat{N}_{L,LS} - i + 1)(\hat{\beta}_{L,LS} + t_{i-1})^2}{[(\hat{N}_{L,LS} - i + 1)\hat{\alpha}_{L,LS} - 1]^3} \tag{4.598}$$

To implement this model, the data required are:

Data Requirement

The times between error occurrences, i.e., the X_i's, or the times of error occurrences, i.e., the t_i's where

$$X_i = t_i - t_{i-1}. \tag{4.599}$$

Once the parameters N, α, and β are estimated, all of the previous quantities developed in this paragraph can be estimated by replacing the parameters with their corresponding estimates.

An alternate Bayesian modification of the Jelinski-Moranda Model is given in a paper by Littlewood and Sofer.[66] In that paper, the times between error occurrences, i.e., X_i's, are assumed exponential, but with parameters

$$\lambda_i = \lambda - (i - 1)\phi \qquad i = 1,\ldots,n. \tag{4.600}$$

Constrast this with the model considered in this paragraph of

$$\lambda_i = \phi_1 + \phi_2 + \ldots + \phi_{N-i+1} \tag{4.601}$$

For both formulations, the λ_i's are taken as random variables. For the alternate model, the λ and ϕ are taken as independent random variables with prior distributions Gamma(b,c) and Gamma(f,g), respectively. All of the quantities developed in this section for the first Bayesian Model are developed for the analogous ones in Littlewood and Sofer's report. They are not repeated here.

4.3.2 Littlewood and Verrall's Bayesian Reliability Growth Model

The next model considered is the Bayesian Reliability Growth Model proposed by Littlewood and Verrall.[67,68,69] The model tries to account for error generation in the corrective process by allowing for the probability that the program could be worsened by correcting the error. The intention is to make a program more "reliable" when an error is discovered and corrected, but there is no assurance that this goal is achieved. With each error correction, a sequence of programs is actually generated. Each is obtained from its predecessor by attempting to correct an error. Because of the uncertainty involved in this correction process, the relationship that one program has with its predecessor

cannot be determined with certainty. This is a second source of uncertainty in the modeling of software reliability (the first dealing with the variation of the input to the program). The specific assumptions for the model are:

Model Assumptions

(a) Successive execution times between failures, i.e., X_i, i=1,...,n, are independent random variables with probability density functions

$$f(X_i|\lambda_i) = \lambda_i e^{-\lambda_i X_i} \qquad X_i > 0 \quad . \tag{4.602}$$

That is X_i is assumed exponential with parameter λ_i.

(b) The λ_i's form a sequence of independent random variables each with a gamma distribution of parameters α and $\psi(i)$, i.e.,

$$g(\lambda_i) = \frac{[\psi(i)]^{\alpha} \lambda_i^{\alpha-1} e^{-\psi(i)\lambda_i}}{\Gamma(\alpha)} \qquad \lambda_i > 0 \quad . \tag{4.603}$$

The function $\psi(i)$ is taken to be an increasing function of i that describes the "quality" of the programmer and the "difficulty" of the programming task. A good programmer should have a more rapidly increasing function ψ than a poorer programmer. The ψ function reflects past and future changes in reliability as a growth process.

(c) The software is operated in a similar manner as the anticipated operational usage.

By requiring the function ψ to be increasing, the condition

$$P\{\lambda(j) < \ell\} \geq P\{\lambda(j-1) < \ell\} \tag{4.604}$$

for all j is satisifed. This reflects that it is the intention to make the program better after an error occurs and is corrected, but it cannot be assured that our goal is achieved.

When the two sources of randomness are put together, then

$$f[x_i|\alpha, \psi(i)] = \int_0^{\infty} f(x_i|\lambda_i)\, g(\lambda_i) d\lambda_i \tag{4.605}$$

$$= \int_0^{\infty} \frac{\lambda_i e^{-\lambda_i X_i} [\psi(i)]^{\alpha} \lambda_i^{\alpha-1} e^{-\psi(i)\lambda_i} d\lambda_i}{\Gamma(\alpha)} \tag{4.606}$$

$$= \frac{\alpha\,[\psi(i)]^{\alpha}}{[x_i + \psi(i)]^{\alpha+1}} \qquad x_i > 0. \tag{4.607}$$

Notice that the x_i's are no longer exponential. They now have the Pareto distribution. The joint density for the x_i's is then

$$f\left[x_i,\ldots,x_n \middle| \alpha,\ \psi(i)\right] = \frac{\alpha^n \prod_{i=1}^{n} [\psi(i)]^{\alpha}}{\prod_{i=1}^{n} [x_i + \psi(i)]^{\alpha+1}} \qquad x_i > 0,\ i = 1,\ldots,n\ . \tag{4.608}$$

Littlewood and Verrall suggest the following forms for the ψ function:

$$\psi(i) = \beta_0 + \beta_1 i \qquad \text{(linear)} \tag{4.609}$$

and

$$\psi(i) = \beta_0 + \beta_1 i^2\ . \qquad \text{(quadratic)} \tag{4.610}$$

(Littlewood finds, for one set of data on which the model was applied, that the linear function is superior to the quadratic function.[68]) In either case, the likelihood function is now a function of three unknowns (α, β_0, and β_1). MLEs could be found by finding the $\hat{\alpha}_{LV}$, $\hat{\beta}_{0,LV}$, and $\hat{\beta}_{1,LV}$, which:

$$L\ (\hat{\alpha}_{LV},\ \hat{\beta}_{0,LV},\ \hat{\beta}_{1,LV}) = \max_{(\alpha,\ \beta_0,\ \beta_1)} L(\alpha,\ \beta_0,\ \beta_1) \tag{4.611}$$

where

$$L(\alpha,\ \beta_0,\ \beta_1) = f(x_1,\ldots,x_n \mid \alpha,\ \beta_0,\ \beta_1)\ . \tag{4.612}$$

These MLEs are the solutions to the following system of equations:

Estimates - Maximum Likelihood

$$\frac{\partial L}{\partial \alpha} = \frac{n}{\hat{\alpha}} + \sum_{i=1}^{n} \ell n\ \hat{\psi}(i) - \sum_{i=1}^{n} \ell n\ [x_i + \hat{\psi}(i)] = 0 \tag{4.613}$$

$$\frac{\partial L}{\partial \beta_0} = \hat{\alpha} \sum_{i=1}^{n} \frac{1}{\hat{\psi}(i)} - (\hat{\alpha} + 1) \sum_{i=1}^{n} \frac{1}{x_i + \hat{\psi}(i)} = 0 \tag{4.614}$$

and

$$\frac{\partial L}{\partial \beta_1} = \hat{\alpha} \sum_{i=1}^{n} \frac{i'}{\psi(i)} - (\hat{\alpha} + 1) \sum_{i=1}^{n} \frac{i'}{x_i + \psi(i)} = 0 \tag{4.615}$$

where

$$\psi(i) = \beta_0 + \beta_1 i \text{ or } \beta_0 + \beta_1 i^2 \tag{4.616}$$

and

$$i' = i \text{ or } i^2 \, . \tag{4.617}$$

Littlewood and Verrall eliminate the parameter α through a Bayesian analysis. By assuming a uniform prior for α, it can be shown (Reference 67) that the distribution of x_i is:

$$f(x_i | \beta_0, \beta_1) = \frac{i\gamma^i}{x_i + \psi(i)} \left[\left(\gamma + \ell n \left(\frac{x_i + \psi(i)}{\psi(i)} \right) \right)^{i+2} \right]^{-1} \, . \tag{4.618}$$

The MLEs for β_0 and β_1 are those parameters which then:

$$L(\hat{\beta}_0, \hat{\beta}_1) = \max_{(\beta_0, \beta_1)} L(\beta_0, \beta_1) = \max_{(\beta_0, \beta_1)} \prod_{i=1}^{n} f(x_i | \beta_0, \beta_1) \, . \tag{4.619}$$

Littlewood and Verrall present an alternative way of estimating β_0 and β_1 based upon goodness-of-fit. The reader is referred to their paper[67] for details.

Another procedure for estimation is based upon least squares. Since

$$f[x_i | \alpha, \psi(i)] = \frac{\alpha \, [\psi(i)]^{\alpha}}{[x_i + \psi(i)]^{\alpha + 1}} \, , \tag{4.620}$$

the MTBF is

$$E\{X_i\} = \int_0^{\infty} \frac{\alpha \, x_i [\psi(i)]^{\alpha}}{[x_i + \psi(i)]^{\alpha+1}} \, dx_i \, , \tag{4.621}$$

$$= \frac{[\psi(i)]}{\alpha - 1} \tag{4.622}$$

provided $\alpha > 1$.

The least squares estimates are those parameters which minimize:

$$S(\alpha, \beta_0, \beta_1) = \sum_{i=1}^{n} \left(x_i - \frac{[\psi(i)]}{\alpha - 1} \right)^2 \, . \tag{4.623}$$

In particular, if

$$\psi(i) = \beta_0 + \beta_1 i \tag{4.624}$$

the least squares estimates $\hat{\alpha}_{LS}$, $\hat{\beta}_{0,LS}$, and $\hat{\beta}_{1,LS}$ satisfy the following system of equations:

Estimates - Least Squares

$$\frac{\partial S}{\partial \alpha} = \hat{\alpha}_{LS} - 1 - \frac{\sum_{i=1}^{n} \hat{\psi}^2(i)}{\sum_{i=1}^{n} x_i \hat{\psi}(i)} = 0 \quad , \tag{4.625}$$

$$\frac{\partial S}{\partial \beta_0} = \sum_{i=1}^{n} x_i - \frac{n \hat{\beta}_{0,LS}}{\hat{\alpha}_{LS} - 1} - \frac{\hat{\beta}_{1,LS} n(n+1)}{2(\hat{\alpha}_{LS} - 1)} = 0 \quad , \tag{4.626}$$

and

$$\frac{\partial S}{\partial \beta_1} = \sum_{i=1}^{n} x_i i - \frac{n(n+1)\hat{\beta}_{0,LS}}{2(\hat{\alpha}_{LS} - 1)} - \frac{\hat{\beta}_{1,LS} \sum_{i=1}^{n} i^2}{(\hat{\alpha}_{LS} - 1)} = 0 \, . \tag{4.627}$$

The data required to implement this model are:

Data Requirement

The times between error occurrences, i.e., the x_i's.

4.3.3 Thompson and Chelson's Bayesian Reliability Model

The last model considered in the Bayesian framework for software reliability is one proposed by W. E. Thompson and P. O. Chelson.[70] The model they developed is one step in the direction of obtaining total system reliability. Their ultimate goal of system reliability included system malfunctions not only due to software but to hardware and unknown or ambiguious source-related malfunctions as well. In a paper by R. Haynes and W. E. Thompson,[71] this total system reliability model is formulated. The one aspect of this model that this paper presents is the reliability model developed for the software related errors. This model attempts to account for the fact that a given software program might be error-free (hence, an infinite MTBF)[72] and it provides for software redesign and repair after malfunctions are observed in a given test phase. The specific assumptions for their model are:

Model Assumptions

(a) The program is not corrected during a testing cycle-only at the completion of a cycle and before the start of a new one.

(b) The software is operated in a similar manner as the anticipated operational usage.

(c) The software errors are assumed to occur at some unknown constant rate, λ. The total number of errors observed in a testing cycle of length T follows a Poisson distribution with parameter λT; i.e.,

$$f(f_i|\lambda) = \frac{e^{-\lambda T}(\lambda T)^{f_i}}{f_i!} \qquad f_i = 0,1,... \tag{4.628}$$

(d) If p denotes the probability that the software contains one or more errors, it can be assumed that p has a prior distribution that is beta with parameters a + 1 and b + 1, i.e.,

$$g(p) = \frac{\Gamma(a + b + 2)}{\Gamma(a + 1)\Gamma(b + 1)} p^a(1 - p)^b \quad 0 \leq p \leq 1 \quad a, b \geq -1. \tag{4.629}$$

The parameter a is thought of as the number of previously delivered software packages with errors among a total of a + b delivered.

(e) The uncertainty about the parameter λ is expressed as a prior distribution for λ. It is assumed gamma with parameters T_0 and f_0+1; i.e.,

$$h(\lambda) = \frac{T_0(\lambda T_0)^{f_0}}{\Gamma(f_0 + 1)} \exp(-\lambda T_0) \qquad \lambda > 0 \ . \tag{4.630}$$

The f_0 can be thought of as the number of software-related system malfunctions in previous testing of total duration T_0.

Thompson and Chelson consider two situations. One is the situation when it is known before testing begins that the software contains errors, i.e., p = 1. The other situation is when there is uncertainty about whether the software does or does not contain errors. This is expressed by the use of the prior in assumption (d). If in this latter situation, an error is discovered in the testing cycle, p is set equal to 1 and the prior g(p) is made a Dirac delta function at p = 0.

For the first situation, Thompson and Chelson show that if f_i errors are observed in testing time T_i, then the posterior distributions for λ and R (the software reliability), are:

$$h(\lambda \mid f_i) = \frac{(T_i + T_0)^{f_i+f_0+1} \lambda^{f_i+f_0}}{\Gamma(f_i + f_0 + 1)} \exp\left[-\lambda(T_i + T_0)\right] \quad \lambda > 0 \qquad (4.631)$$

and

$$f(R \mid f_i) = \left(\frac{T_i + T_0}{t}\right)^{f_0+f_i} \cdot \left[\ell n \left(\frac{1}{R}\right)\right]^{f_0+f_i-1} \frac{R^{\left(\frac{T_i + T_0}{t} - 1\right)}}{\Gamma(f_0 + f_i)} \qquad (4.632)$$

$0 \leq R \leq 1$.

(The t is the postulated mission time for the program.) The distribution for R reflects the posterior view of the program reliability after f_i errors are observed in the current testing cycle.

The second situation is the one in which no errors are observed in the testing cycle i; i.e., $f_i = 0$. This generates the uncertainty about whether the program does or does not have any errors still residing in the code. For this situation, the posterior cumulative distribution functions for λ and R are shown to be:

$$H(\lambda \mid f_i = 0) = \frac{a + 1}{a + b + 2} \int_0^{\lambda} (T_i + T_0)^{f_0+1} x^{f_0} \exp\left(-x(T_i + T_0)\right) dx + \left(\frac{b + 1}{a + b + 2}\right) \qquad (4.633)$$

and

$$F(R \mid f_i = 0, p) = p \int_0^{R} \frac{(T_i + T_0)^{f_0 + 1} \left[\ell n \frac{1}{x}\right]^{T_i+T_0-1}}{\Gamma(f_0 + 1)} dx \quad 0 \leq R < 1 \qquad (4.634)$$

$$= 1 \qquad R = 1. \qquad (4.635)$$

If a squared error loss function is assumed in estimating λ and R, the Bayes' estimates for λ and R are then the means of the respective posterior distributions. They are shown[71] to be:

Estimates - Bayes

$$\hat{\lambda} = \frac{(a + 1)}{(a + b + 2)} \cdot \frac{(f_0 + 1)}{(T_i + T_0)} \qquad (4.636)$$

and

$$\hat{R} = \frac{a + 1}{a + b + 2} \left(\frac{T_i + T_0}{T_i + T_0 + 1} \right)^{f_0+1} + \frac{b + 1}{a + b + 2} \tag{4.637}$$

for $f_i = 0$ and

$$\hat{\lambda} = \frac{f_i + f_0 + 1}{T_i + T_0} \tag{4.638}$$

and

$$\hat{R} = \left(\frac{T_i + T_0}{T_i + T_0 + t} \right)^{f_0+f_i} \tag{4.639}$$

for $f_i > 0$.

Thompson and Chelson show, in their paper, how the various distributions can be used in determining when testing is to be terminated. The decision rule to do this is formulated in a sequential manner. The reader is referred to their paper for additional details.

Notice that if i - 1 test periods have elapsed, then

$$T_0 = \sum_{i=1}^{i-1} T_i \tag{4.640}$$

and

$$f_0 = \sum_{i=1}^{i-1} f_i \ . \tag{4.641}$$

The data required to implement their model are:

Data Requirements

(a) The number of software errors discovered in each period of testing, i.e., f_i's.

(b) The length of testing time for each period, i.e., the T_i's.

(c) For the total number of software packages that have been released, the number found to contain errors. These numbers are used in determining the prior distribution for p at any stage.

4.4 MARKOV MODELS

This next class of models views the software correction process as a discrete space system in which a transition from one state to another occurs whenever an error detection or correction is made. These models attempt to achieve a more precise and realistic error behavior prediction but at the cost of a great deal of added complexity. In fact, the models in many cases cannot be used to derive a closed form solution. Only large sample approximations can be given or approximate numerical solutions can be stated. Much research is still needed in this area of modeling. This section is included for completeness in the presentation of the various approaches to software reliability modeling. Because of the complexity of the models, this section is not developed in great detail. The reader is referred to the respective research articles for additional details.

4.4.1 Trivedi and Shooman's Many State Markov Models

The basic model and its generalizations are presented in a paper by Ashok Trivedi and Martin Shooman[73] under contract to the Office of Naval Research and the Rome Air Development Center. The model is used in providing estimates of the reliability and availability of a software program based upon an error detection and correction process. Availability is defined as the probability that the program is operational at a specified time. The software can be viewed in either one of two states "up" or "down." The system is in an up state if no errors have occurred or an error has just been corrected. The software is in a down state when an error has been discovered and is being corrected. The sequence of up state is denoted by $(n, n-1, n-2, \ldots, n-k, \ldots)$ while the sequence of down states is denoted as $(m, m-1, \ldots, m-k, \ldots)$. The system is in the up state, $n-k$, if the $(k-1)$st error has been detected but the kth has not. It is in the down state, $m-k$, if the kth error has been detected but not yet corrected.

The specific assumptions for the model are:

Model Assumptions

(a) The transition probability from state i to state j (p_{ij}) is dependent only on those states and is independent of all past states except the last one. (The Markov property.)

(b) The error detection rate for the state $n-k$ is known; denote it as λ_{n-k}. The error correction rate for state $m-k$ is known; denote it as μ_{m-k}.

(c) Finite nonzero times are spent by the system only in the system states. The transition times are infinitesimally small so the probability of two or more error detections or corrections within this time frame is zero.

(d) The software is operated in a similar manner as the anticipated operational usage.

(e) When a software error is corrected, it is done without the introduction of additional errors.

(f) The program is assumed to be fairly large (the order of 10^5 words or more of code).

Some of the generalizations of this basic model include allowing error introduction in the correction process and the system can be in more than two states. A generalization of the basic model allows a third state, a "noncritical down" state. The reader is referred to Trivedi and Shooman's paper for details.

The two specific cases of the basic model that this pap. reviews are called Model I and Model II. For Model I, the error detection rate, λ_{n-k}, and the error correction rate, μ_{m-k}, are taken as functions of the number of errors that have occurred, i.e., k. In Model II, the rates are taken as functions of time. The choice between the two models is determined by the way the error data are collected. For Model I, the individual errors are recorded along with the time of occurrence of each error. For Model II, the number of errors is recorded over the operating time of the program.

For either model, the derivation of the reliability and availability is as follows. Suppose $P_{n-k}(t)$ denotes the probability that at time t the state is in the n - k up state. Similarly, $P_{m-k}(t)$ is the corresponding probability, the system is in the m - k down state. Then the availability of the program is:

$$A(t) = P\{\text{system is up at time } t\} \tag{4.642}$$

$$= P_n(t) + P_{n-1}(t) + \ldots \tag{4.643}$$

$$= \sum_{k=0}^{\infty} P_{n-k}(t) \; . \tag{4.644}$$

Thus, only the probabilities for the various up states for the system are needed to derive the availability. The reliability on the other hand, depends upon the stage of debugging since the smaller the number of residual errors, the less likely it is for the program to "discover" them. Suppose the system has just entered the state n - k at time t. Suppose this time is renamed as $\tau = 0$, then in the interval $(0, T_K)$, where T_k is the time of discovery of the kth error, the error occurrence rate, $\lambda(k)$, is a constant. The reliability function is then

$$R(\tau) = e^{-\lambda(k)\tau} \qquad 0 \leq \tau \leq T_k, \qquad k = 1,2,\ldots. \tag{4.645}$$

Hence, after the (k - 1)st error has been corrected, only $\lambda(k)$ is needed to establish the reliability for the program for all times between the occurrence of the (k - 1)st and the kth error.

Now consider how the state probabilities are derived. First, Model I is considered with the special case of constant error detection and correction rates, i.e.,

$$\lambda_{n-k} = \lambda_{n-k}(k) = \lambda \qquad k = 0,1,. \tag{4.646}$$

and

$$\mu_{m-k} = \mu_{m-k}(k) = \mu \qquad k = 0,1,.... \tag{4.647}$$

For any Δt (Δt small), the following system of equations represents the transition behavior of the Markov system:

$$P_n(t + \Delta t) = (1 - \lambda\Delta t)P_n(t), \tag{4.648}$$

$$P_{n-k}(t + \Delta t) = (1 - \lambda\Delta t)P_{n-k}(t) + \mu\Delta t\, P_{m-k+1}(t) \qquad k = 1,2,... \tag{4.649}$$

and

$$P_{m-k}(t + \Delta t) = (1 - \mu\Delta t)P_{m-k}(t) + \lambda\Delta t P_{n-k}(t) \qquad k = 0,1,... \tag{4.650}$$

By dividing both sides of the previous equations by Δt and letting $\Delta t \to 0$, the following set of differential equations is obtained:

$$\dot{P}_n(t) = -\lambda P_n(t). \tag{4.651}$$

$$\dot{P}_{n-k}(t) + \lambda P_{n-k}(t) = \mu\, P_{m-k+1}(t) \qquad k = 1,2,...$$

and

$$\dot{P}_{m-k(t)} + \mu\, P_{m-k}(t) = \lambda\, P_{n-k}(t) \qquad k = 0,1,2... \tag{4.652}$$

Using the initial conditions:

$$P_n(0) = 1 \quad ,$$

$$P_{n-k}(0) = 0 \qquad k = 1,2,3,...$$

and

$$P_{m-k}(0) = 0 \qquad k = 0,1,2,...$$

Trivedi and Shooman[73] show that the solutions to this system of equations are:

$$P_{n-k}(t) = \left(\frac{\mu\lambda}{\mu - \lambda}\right)^k e^{-\lambda t} \sum_{j=0}^{k} \frac{1}{(\mu - \lambda)^j} \frac{t^{k-j}}{(k - j)!}\left\{(-1)^{k+1} c_{kj} e^{-(\mu-\lambda)t} + (-1)^j d_{kj}\right\} \qquad k = 0,1,2,... \tag{4.653}$$

where the constants $\{c_{kj}\}$ and $\{d_{kj}\}$ are given by:

$$c_{k0} = 0,\ c_{k1} = 1,\ d_{k0} = 1 \quad ;\ k = 0,1,2,\ldots, \tag{4.654}$$

$$c_{kj} = \binom{k+j-1}{j-1} \quad k = 2,3,\ldots \quad ;\ j = 2,3,\ldots,\ k\ , \tag{4.655}$$

and

$$d_{kj} = \binom{k+j-1}{j-1} \quad k = 1,2,\ldots \quad ;\ j = 1,2,\ldots,\ k\ , \tag{4.656}$$

and

$$P_{m-k}(t) = \frac{1}{\mu}\left(\frac{\mu\lambda}{\mu - \lambda}\right)^{k+1} e^{-\lambda t} \sum_{j=0}^{k} \frac{\gamma_{k,j+1}}{(\mu - \lambda)^j} \cdot \frac{t^{k-j}}{(k - j)!} \cdot$$

$$\left\{(-1)^j + (-1)^{k+1} e^{-(\mu-\lambda)t}\right\} \quad ;\ k = 0,1,2,\ldots \tag{4.657}$$

where the constants $\{\gamma_{k,j+1}\}$ are given by:

$$\gamma_{k,1} = 1 \quad ;\ k = 0,1,2,\ldots \tag{4.658}$$

$$\gamma_{k,j} = \binom{k+j-1}{j-1}; \quad k = 1,2,3,\ldots \quad j = 2,3,\ldots,k+1\ . \tag{4.659}$$

In the general case of Model I, where the rates are assumed functions of the number of errors discovered, the previous set of differential equations becomes:

$$\dot{P}_n(t) = -\lambda(0)\ P_n(t) \tag{4.660}$$

$$\dot{P}_{n-k}(t) + \lambda(k)\ P_{n-k}(t) = \mu(k-1)\ P_{m-k+1}(t) \qquad k = 1,2,3,\ldots \tag{4.661}$$

and

$$\dot{P}_{m-k}(t) + \mu(k)\ P_{m-k}(t) = \lambda(k)\ P_{n-k}(t) \quad ; \qquad k = 0,1,2,\ldots \tag{4.662}$$

under the same initial conditions given before. Trivedi and Shooman recommend that this system of differential equations be solved numerically using the Runge-Kutta Approach.

For Model II, if the rates are constant, exactly the same solution as obtained for Model I with constant rates is obtained. For the general case, i.e.,

the rates considered as functions of time, there is an analogous set of differential equations as obtained in Model I; namely,

$$\dot{P}_n(t) = -\lambda(t)\, P_n(t) \quad , \tag{4.663}$$

$$\dot{P}_{n-k}(t) + \lambda(t)\, P_{n-k}(t) = \mu(t)\, P_{m-k+1}(t) \quad k = 1,2,\ldots, \tag{4.664}$$

and

$$\dot{P}_{m-k}(t) + \mu(t)\, P_{m-k}(t) = \lambda(t)\, P_{n-k}(t); \quad k = 0,1,2,\ldots. \tag{4.665}$$

The initial conditions are the same as those as given in Model I.

The solutions to this system of equations may not even exist in closed form. As for Model I, numerical solutions must therefore be relied upon. Trivedi and Shooman's paper is referred to for additional details.

The data required to implement this model are:

Data Requirements

(a) The error detection rate between the times of error occurrence which is either expressed as a function of the number of errors detected or as a function of time.

(b) The error correction rate between the times of error occurrence which is expressed either as a function of the numbers of errors detected or time.

The interesting aspect of this model, aside from the Markovian aspect, is that no parameters are estimated. The error detection and correction rates are needed as input into the model formulation. If these are unknown, which is the usual situation, they need to be estimated. In the example application considered by Trivedi and Shooman, empirical estimates of the rates are obtained as a function of time by using the number of software error reports per month for the detection rate and the number of closed software error reports per month for the correction rate. Sukert[17] uses the number of errors found and corrected per day as reported in software error forms to estimate the rates in this application. It employs standard regression analysis as well as fitting some nonlinear functions to the data to estimate the curves $\lambda(t)$ and $m(t)$.

4.4.2 Littlewood's Semi-Markov Model

The last model considered in this paragraph was proposed by Bev Littlewood.[74, 75] The model incorporates the structure of the program in developing its availability. One of the major weaknesses of the previous time-dependent models is that the structure of the program is not considered in determining its reliability. Littlewood adopts a modular approach to the software and attempts to

describe this structure via the program's dynamic behavior using a Markov assumption. The program comprises a finite number of modules with exchanges of control between them that follow a semi-Markov law. The time spent in a given module can be taken as a random variable with any distribution (hence, semi-Markov) which is characteristic of the module and the module that it transitions to. The specific assumptions of this model are:

Model Assumptions

(a) The program is composed of M modules. Transitions between modules are such that the probability that the program terminates one module to enter another is independent of the time the first module is entered (semi-Markov property).

(b) When the program is in module i, the failures are assumed to follow a Poisson process with parameter v_i.

(c) When module i calls module j with probability p_{ij}, the probability of a failure in the interface between the two modules is α_{ij}.

(d) The distribution of the time spent in module i before entering module j depends upon only i and j and is known only via the first two moments $\mu_1{}^{ij}$ and $\mu_2{}^{ij}$.

(e) The program is operated in a similar manner as the anticipated operational usage.

(f) Each failure results in a random variable cost. The random variables are assumed independent with distributions dependent on the module or interface in which the failure occurs. The distributions are only known through their first two moments.

The last assumption is optional; it is only needed if an overall failure cost analysis is needed.

Suppose N(t) is the total number of failures (both within modules and between) observed in the program in the time interval (0,t). Deriving the distribution of N(t) for a specified t is an extremely difficult if not impossible task. A complete description of the behavior of N(t) requires knowledge of the distributions of times within modules, a requirement that is usually unattainable in practice. Littlewood derives an asymptotic result pertaining to the behavior of N(t). If the very plausible assumption for a modular program is made that the individual failure rates are much smaller than the switching rates between modules, then the failure point process of the integrated program is asymptotically a Poisson process with rate parameter

$$\frac{\sum_{i=1}^{M}\sum_{j=1}^{M}\Pi_i p_{ij}\,(\mu_1{}^{ij}v_i + \alpha_{ij})}{\sum_{i=1}^{M}\sum_{i=1}^{M}\Pi_i p_{ij}\mu_1{}^{ij}} \tag{4.666}$$

as v_i, $\alpha_{ij} \to 0$ where $\underset{\sim}{\Pi} = \{\Pi_i\}$ satisfies $\underset{\sim}{\Pi} \cdot P = \underset{\sim}{\Pi}$ with $\sum_{i=1}^{M} \pi_i = 1$ and P is the MxM transition matrix of the system.

The interesting aspect of this result is demonstrated by rewriting the previous expression; i.e., if

$$a_i = \frac{\sum_{j=1}^{M}\Pi_i p_{ij}\mu_1{}^{ij}}{\sum_{i=1}^{M}\sum_{j=1}^{M}\Pi_i p_{ij}\mu_1{}^{ij}} \tag{4.667}$$

and

$$b_{ij} = \frac{\Pi_i p_{ij}}{\sum_{i=1}^{M}\sum_{j=1}^{M}\Pi_i p_{ij}\mu_1{}^{ij}} \tag{4.668}$$

then the previous expression can be reexpressed as:

$$\sum_{i=1}^{M} a_i v_i + \sum_{i=1}^{M}\sum_{i=1}^{M} b_{ij}\alpha_{ij} \quad . \tag{4.669}$$

The a_i represents the limiting proportion of time spent in module i while b_{ij} is the limiting frequency of i to j module transfers. It is often possible to estimate them directly. An extremely complex description of the behavior of N(t) is therefore represented asymptotically in a very simplistic manner.

Suppose interest is also in a failure cost analysis. If $Y_i(t)$ represents the random variable for the cost of a failure in module i, and $Y_{ij}(t)$ represents the random variable cost of a failure in transfer from module i to j, the total program cost is:

$$Y(t) = \sum_{i=1}^{M} Y_i(t) + \sum_{\substack{i=1 \\ i\neq j}}^{M}\sum_{j=1}^{M} Y_{ij}(t) \quad . \tag{4.670}$$

Again the exact description of Y(t) is extremely complex if not impossible to develop. Littlewood develops an asymptotic result which depends only upon the means and variances of the defining distributions. He shows that:

$$\frac{Y(t) - \mu t}{\sigma t^{\frac{1}{2}}} \xrightarrow{\mathcal{L}} N(0,1) \tag{4.671}$$

where μ is the (asymptotic) mean cost incurred per unit time for the integrated program and is:

$$\mu = \frac{\sum_{i=1}^{M} \sum_{j=1}^{M} \Pi_i p_{ij} \left(\mu_1{}^{ij} v_i \mu_1{}^{i} + \alpha_{ij} \mu_1{}^{ij}\right)}{\sum_{i=1}^{M} \sum_{j=1}^{M} \Pi_i p_{ij} \mu_1{}^{ij}} \tag{4.672}$$

where $\mu_1{}^{i}$ and $\mu_1{}^{ij}$ are the means of Y_i, Y_{ij} respectively.

The reader is referred to Littlewood's papers for additional details pertaining to this result and a definition of σ.

The major problem with this model, as with the previous case is that all of the parameters that make up the model are input; therefore, they must be known or estimated. The data required for this model are:

Data Requirements

(a) The transition probabilities from modules i to j, i.e., the p_{ij}'s.

(b) The error rates within the modules i, i.e., the v_i's.

(c) The first two moments of the distribution for the time spent in module i before transferring to module j, i.e, the $\mu_1{}^{ij}$'s and $\mu_2{}^{ij}$'s.

(d) The probabilities of failures occurring at the interfaces between modules, i.e., the α_{ij}'s.

(e) If a cost analysis is desired, the first two moments of the cost distributions of failure within and between modules.

As can be seen, the data required can easily prohibit the use of this model. An additional factor to consider before applying this model (and the previous one) is assuring that the Markov property is satisfied. This could prove the most formidable problem of all in applying Markov Models.

CHAPTER 5

COMPARISON OF RELIABILITY MODELS

For this part of the report, some studies and the results pertaining to comparing the performance of the various reliability approaches and models on software error data sets are described. Any common conclusions that have been reached among the studies are pointed out. The studies described all involve the comparison of at least three or more software reliability models. Excluded are various individual studies that have been done on a given software model only. This includes the results of applying Musa's Model to error data sets [54,55] and Littlewood's Bayesian Model. Musa's Model has been applied to actual software error data, with some success. Littlewood also has applied his Bayesian Model to some data sets with a good match obtained between the predicted and actual error observations. However, this comparison is strictly concerned with results that can be given in the performance of one reliability model in contrast to another. From these studies, some guidelines can be established for employing a model in a given situation. In this paragraph, the analysis is limited to the Time Domain Approach rather than Error Seeding/Tagging and the Data Domain Approaches since no major studies have been done to compare the performance of the different approaches. The first major effort was made by Alan Sukert of the Rome Air Development Center.[17,76,77] The study involves five major models: The Jelinski-Moranda Model (Paragraph 4.2.3), the Schick-Wolverton Model (Paragraph 4.2.4), a modified Schick-Wolverton Model (Paragraph 4.2.4.1), and Geometric (Paragraph 4.2.6) and Modified Geometric Models (Paragraph 4.2.6.1). Other models were considered but, due to data requirements for these models (e.g., CPU time), could not be used. The chosen models were applied to four large scale DOD software projects. Using the software error reports filed for each of the projects, the error counts per day and per week were used as input into the software models. (All models were modified to allow more than one error per time frame.) The study considered estimates for the respective models obtained from maximum likelihood and least square procedures.

The basic conclusions drawn from this comparative study are:

(a) The grouping by weeks does better in predictive ability than by the day.

(b) The Jelinski-Moranda and Schick-Wolverton Models give reasonable predictions for small projects while the modified Schick-Wolverton Model does better for larger ones.

(c) The Geometric Models are better to use when the MTBF or reliability estimates are of concern.

A major problem experienced in the application of these models in this study (as in the others) is the problem of convergence. The estimation procedures failed

in many instances to come up with model parameter estimates. This is discussed in some detail after the next study is described.

The second major comparative effort was undertaken by Hughes Aircraft Company.[33] The study involves the Generalized Poisson Model of Paragraph 4.2.5 with $g(X_1,\ldots,X_n) = X^{\alpha}$; ($\alpha$ unknown), a binomial model and the Nonhomogeneous Poisson Process Model of Paragraph 4.2.9. The various models employ both maximum likelihood and least squares in the estimation of model parameters. These models are applied to 16 sets of electronic's system computer program software data. The major conclusions reached are:

(a) Generally, the model fits to the data are poor, but the best fitting of the three and the one applicable to the most data sets is the Generalized Poisson Model.

(b) Grouping the error data by a time period has better convergence properties than ungrouped.

(c) Maximum likelihood and least squares estimates for a given model are similar.

The major problem of lack of convergence to parameter estimates was experienced in this study as well. The authors suspect, as does Sukert in the previous study, that a major problem causing this lack of convergence is violation of the assumptions on which the models are based, especially the violation of a nonincreasing error rate. The Hughes report finds that by plotting the estimated error rate whenever it is increasing in a region is precisely the region in which convergence problems are experienced. The types of convergence problems encountered include lack of convergence, oscillation, convergence to a nonoptimal solution, and nonuniqueness of the solution. These problems are especially experienced by the MLEs. The report employs a second derivative criterion to weed out nonoptimal solutions, but the report points out that this cannot be relied on completely because of computer precision problems in finding the optimal solutions.

The third study was undertaken by Dayton University under contract to the Air Development Center.[18] The study applied the Jelinski-Moranda Model, the Geometric Model, the extended Jelinski-Moranda Model (Paragraph 4.2.3.2), and Schneidewind's Model with approach (c) (Paragraph 4.2.8) to software error data. The first two are applied to two data sets in which the times between error occurrences are recorded while the latter two are applied to two data sets in which the error counts are recorded. The conclusions are:

(a) If the times between error occurrences are available, the Geometric Model does a better predictive job, but if the error counts per time interval are available, the Schneidewind Model is preferred.

(b) The extended Jelinski-Moranda Model and Schneidewind's Model give similar results, but the extended Jelinski-Moranda Model is very sensitive to changes in the data.

(c) The Jelinski-Moranda Model tends to estimate a smaller number of errors remaining than the Geometric Model, illustrating the "optimistic" tendency of the exponential class of distributions.

The fourth study was performed by the University of Utah.[32] In contrast to the previous three studies, this study compares a number of models using deterministically generated error data rather than actual data. The times of error occurrences are generated on a computer, following the underlying model with known model parameters. The models considered include: the Jelinski-Moranda Model, the Geometric Model, and Musa's Model. A Monte-Carlo study of the behavior of the least squares and MLEs was undertaken. The results of this study are:

(a) A strong positive correlation is indicated among the various estimates for total number of errors,

(b) The estimate of the MTBF is best for the Geometric Model,

(c) The accuracy of the estimates increases as either the total number of errors increase or the number of errors remaining decreases, and

(d) The least squares estimation, using the times between error occurrences rather than the actual times of the occurrences, does not perform as well as the other estimators.

The last few paragraphs summarize the results of the various studies undertaken to compare the various models. As can be seen from the studies, additional comparative research is needed. Many of these studies employed error data that were gathered without the data requirements or assumptions of the various models in mind. What is needed is a large scale effort in which the data are gathered under a controlled environment. Currently such a study involving the Nonhomogenous Poisson, the IBM Poisson Model, the Generalized Poisson Model, the Jelinski-Moranada Model, and the Geometric Poisson Model is being undertaken by Hughes Aircraft for the Rome Air Development Center. The data are being collected specifically for software reliability applications. Although the final report is not written yet, an interim report (Reference 78) finds some results similar to the previous studies. Specifically, the major problem of convergence and the violation of the model assumptions are found. Again the major violation is an error rate that is nonconstant during a testing interval and nondecreasing over all intervals.

It is difficult, based upon the results of the studies, to provide clear cut guidelines in applying the software models. We can only conjecture. Out of the various models considered, it appears that the Generalized Poisson or Schneidewind's Model approach (c) might be best suited for count data. The Geometric Models should be considered when estimating MTBF. The convergence problem appears to diminish as the length of the testing period increases. However there is no method to determine what the optimal length of a testing interval should be. This depends upon the underlying error generation process which is not known.

Many of the models discussed in this report have yet to be compared on the basis of performance with others. None of the Bayesian Models, Markov Models, the Error Seeding/Tagging Models, or the Data Domain Approach to software reliability modeling have been included in any comprehensive study. (Note: Hughes' current research will incorporate Littlewood's Bayesian Model). This report can only present what has been done and what those limited results indicate. Much is yet to be done-if it even can be done. A large scale controlled-data collection, in which the CPU time and wall clock time are simultaneously gathered for the purpose of comparing as many different models as possible, may be economically and administratively infeasible. Moreover, for the modeling of software error generation, no one model is applicable in all instances. The software analyst needs a collection of software models which have demonstrated themselves in various environments and comparative studies. From this collection, the analyst judiciously selects the one most applicable to his/her situation. Flexibility and adaptability are the keys to successful modeling.

CHAPTER 6

"QUICK" ESTIMATES OF SOFTWARE RELIABILITY MEASURES

This last paragraph briefly presents some proposed "quick" estimates for various software reliability measures. These procedures do not require the extensive error data base of the previous sections. The view taken toward software reliability is very simplistic and pragmatic in nature. The two procedures discussed are not advocated in this report, but are included for completeness. The purpose of this report is to review all of the various procedures that are advocated in determining the reliability status of a set of software.

6.1 MTBF ESTIMATION

This very simple measure of MTBF was proposed by Gregory Hansen of Systems Engineering Laboratories.[79] When a software program is first released, there are only a few users and hence the failure generation is a minimum. This means that the MTBF is fictitiously high, giving the software manager a false sense of security. As the software begins to be used, the MTBF can be expected to rise slightly as the initial gross errors are discovered and eliminated. However, in later years as more and more users test the software, the MTBF drops significantly. Finally, the software reaches a "mature" state and the MTBF increases sharply.

This behavior is reflected in the following formula:

$$MTBF_{(i)} = \frac{(N_i + N_{i-1} + \ldots + N_{i-I+1})*C}{M_i} \qquad (6.1)$$

where M_i = number of software errors discovered in year i,

$MTBF_{(i)}$ = MTBF for year i,

N_i = Number of copies of the program in use for year i,

I = Number of years the program has been used,

and C = Estimates of the average number of hours that the product is used in a year.

An example calculation is as follows. Suppose the MTBF is desired for year i = 3 with the software being distributed to users for 2 years. During the third year, there has been a total of 10 users and during the second, a total of 5 users. The estimated average number of hours that the program is used during the third year

is 500 hours. During the third year, a total of 100 errors have been observed. The formula

$$MTBF_{(3)} = \frac{(N_3 + N_2)*C}{M_3} = \frac{(10 + 5)*500}{100} = 75 \text{ hrs} \qquad (6.2)$$

reflects the total estimated number of hours of use by all users during the given year, divided by the number of errors found during the year. The function can be plotted against time to see the error behavior of the software package. Once the program reaches maturity, future values of MTBF can be predicted by extending the curve.

6.2 PRAGMATIC SOFTWARE RELIABILITY ESTIMATION

The last method considered is proposed by John Wall and Paul Ferguson.[80] Using the basic premise that the failure rate of software decreases as more software is used and tested, they formulate a relationship between the number of failures and the "maturity" of the software. Specifically, the relationship proposed is:

$$C = C_0 \left(\frac{M}{M_0}\right)^{\alpha} \qquad (6.3)$$

where C is the cumulative number of errors experienced for a software program of maturity M. C_0 and α are constants determined empirically by plotting the cumulative number of errors versus the maturity level of the software. M_0 is a scaling constant. Typically, the units of M and M_0 are expressed as: amount of calendar time expended, processor or CPU time, man-months of testing, or the number of tests executed.

The failure rate, R, is then determined as

$$R = \frac{dC}{dt} = \alpha C_0 \frac{d(M/M_0)}{dt} \left(\frac{M}{M_0}\right)^{\alpha-1} \qquad (6.4)$$

For convenience this is expressed as:

$$= R_0 \left(\frac{M}{M_0}\right)^{\alpha-1} \qquad (6.5)$$

where R_0 is simply a constant. Again the terms R_0 and α can be determined empirically from the data. For example, failures per CPU second can be plotted versus number of CPU seconds of operation to determine R_0 and α. Care must be taken to ensure consistency of the units in the functional relationships.

The application of this method is applied to a number of data sets in their paper. The reader is referred to that paper for additional details.

CHAPTER 7

SUMMARY AND CONCLUSIONS

With the ever increasing role that software is playing in the weapon systems and the increased complexity of the programs because of that role, a dramatic increase in the cost of the software over the life cycle of the weapon system is seen. Greater emphasis has thus been placed in determining more cost effective ways of software development and testing. One such method that has developed over the last 10 years is the calculation of a software's reliability. By having a quantative measure of a program's or a program module's reliability, a software manager can best determine the allocation of testing personnel and just how much testing to employ before release to the user.

This report provides a review of the various approaches to estimating that reliability. The three major approaches are categorized as: Error Seeding/Tagging, Data Domain, and Time Domain. The Error Seeding/Tagging Approach uses the concept of error introduction into the software. Based upon the number of inserted errors and inherent errors found in the testing phase, the total number of errors still residing in the program can be estimated. The major problem with this approach is the implementation. How are errors of the same nature and distribution as the inherent errors inserted into a program? The Data Domain Approach bases the reliability estimation on the number of successful execution runs out of the total number of runs attempted. In addition, the approach tries to incorporate the input domain structure into the estimation process. The input space is broken down into regions which are assigned probabilities based upon anticipated operational profiles. Random samples from the input space are then drawn according to these probabilities and the count of successful runs made from them are used in the reliability calculation. The major weakness with this approach is the stratification of the input space and the resulting probability assignments.

The last approach, which this paper deals with the most, is the Time Domain. This approach attempts to model the error generation process as observed over time (either CPU or wall clock). This is done using the time of error occurrence (or equivalently, the time between) or the number of errors observed over a testing interval. Many of the models are based upon an underlying Poisson process for the error generation over a specified time frame or an exponentially distributed random variable for the time between error occurrences. The Time Domain Approach can itself be categorized into three types of models: "Classical," "Bayesian," and "Markov." The "Classical" Models can be traced back to their origin within hardware reliability theory. Many of the concepts of hardware reliability theory (MTBF, hazard rate, reliability function) are adapted to the field of software. Moreover, models of this class tend to view the errors inherent in a program to be of the same order of magnitude and the correction of any one of them has the same order of impact upon the program. The "Bayesian" viewpoint takes this impact and treats it as a random variable. It is not known what effect the correction of an

error might have upon the behavior of the program. When errors are discovered early in the testing cycle, it is expected that the most dramatic improvement in the performance of the program occurs after their correction. Errors discovered late in the cycle have the least dramatic improvement. The "Markov" Models attempt to formulate the error generation process over time as a Markov process in which transition probabilities are either given or derived. These probabilities are the state transition probabilities for moving from one state to another.

As in the previous approaches, the Time Domain also has its share of problems. The Markov Models are extremely complex and difficult to apply. Most of the results are either for special cases or are asymptotic in nature. The Bayesian Models represent a more realistic approach to modeling the actual error generation/correction process. The difficulty here, as with the Bayesian theory in general, is the specification of a prior distribution for the error rate. In addition, little has been done in comparing this class of models to models of a "classical" nature. The major weakness for the Classical Models is an oversensitivity to the violations of the assumptions upon which they rest. They are especially sensitive to an increasing failure rate within the data. This increasing rate may be due to many reasons: introduction of new errors in the correction process, nonuniform testing, and nonuniform application of testing manpower throughout the testing cycle. The last two are especially common occurrences in typical software testing programs. Another problem that these models face is a lack of independence among the errors. In many instances, the discovery of one error quickly leads to others generating a "clumping" effect of the errors over time. These various violations lead to poor fits of the models and convergence problems in the estimation process.

Various studies have been undertaken to compare the performance behavior among the models, but no clear superior model has arisen. It is felt by this author that no one model can be advocated for all applications. A collection of models that have demonstrated themselves over a large class of problems should be considered. The software analyst should then pick from this class the one that is most effective in modeling his/her set of data. Modeling has always been an interative procedure, it includes choosing a candidate model, estimating the parameters of the model, testing the adequacy of the model, and cycling back if necessary.

Much research is yet to be done in this new field; however, software reliability modeling can provide an effective aid to the software manager. Some of the applications of these models demonstrate this. What is to be kept in mind, however, is that it is one of many tools available in developing cost effective software. By careful consideration of the collection of data for these models, to ensure the model assumptions are satisfied as much as possible, and by using a collection of models that appear "robust" to violations of assumptions which cannot be met, the models provide a useful aid. Using the common techniques of modeling, the chosen model can be a useful quantitative measure to determine the length of testing and manpower utilization. Otherwise, if the manager is asked, "What led you to make the decision to release the software?" What can he say?

REFERENCES

1. Shooman, M., "Software Reliability: Measurement and Models," Proceedings of the 1975 Annual Reliability and Maintainability Symposium, 1975, pp. 485-491.

2. Shooman, M., "Software Reliability: Analysis and Prediction," Integrity in Electronic Flight Control Systems, AGARDograph No. 224, Advisory Group for Aerospace Research and Development, Part II, pp. 7, 1977.

3. Air Force Magazine, July 1973, pp. 46-57, Sheeley, Martin, Computer Software Reliability Fact or Myth? Ogden Air Logistics Center Technical Report, AD-780-697, 1973.

4. Mills, H. D., On the Statistical Validation of Computer Programs, FSC-72-6015, IBM Federal System Division, Gaithersburg, Md., 1972.

5. Rudner, B., "Seeding/Tagging Estimations of Software Errors: Models and Estimates," RADC-TR-77-15, Rome Air Development Center, A036655, 1977.

6. Schick, George, J. and Wolverton, Ray W., "An Analysis of Competing Software Reliability Models," IEEE Transactions on Software Engineering, Volume SE-4, No. 2, 1978, pp. 104-120.

7. Basin, S. L., "Estimation of Software Error Rates via Capture-Recapture Sampling," Science Applications, Inc., Palo Alto, California, 1973.

8. Basin, S. L., "Measuring the Error Content of Software," Science Applications, Inc., Palo Alto, Califorinia, 1974.

9. Hecht, Herbert, Measurement, Estimation, and Prediction of Software Reliability, National Aeronautics and Space Administration Technical Report, NASA-CR-145135, prepared by the Aerospace Corporation, El Segundo, Calif., 1977.

10. Brown, J. R. and Lipow, M., "Testing for Software Reliability", Proceedings, 1975 International Conference on Reliable Software, IEEE Catalog No. 75 CH0940-7 CSR.

11. Corcoran, W. J.; Weingarten, H; and Zehna, P.; "Estimating Reliability After Corrective Action," Management Science, Volume 10, No. 4, 1954, pp. 786-795.

12. Nelson, Eldred, "Estimating Software Reliability From Test Data," Microelectronics and Reliability, Volume 17, No. 1, 1978, pp 67-73.

13. Thayer, T. A.; Craig, G. R.; and Frey, L. E.; Software Reliability Study, Rome Air Development Center Technical Report, RADC-TR-76-238, 1976.

14. Sugiura, N.; Yamamoto, M.; and Shiino, T., "On the Software Reliability," Microelectronics and Reliability, Volume 13, 1974, pp. 529-233.

15. Elliott, R. W.; Marchbank, M. P; McWilliams, M. G.; Ringer, L. J.; and Simmons, D. B., "Measuring Computer Software Reliability," Computers and Industrial Engineering, Volume 2, No. 3, 1978, pp. 141-151.

16. LaPadula, L. J., Engineering of Quality Software Systems, Volume VIII - Software Reliability Modeling and Measurement Techniques, Mitre Corp., Rome Air Development Center Technical Report, RADC-TR-74-325, Volume III, Jan. 1975.

17. Sukert, Alan, A Software Reliability Modeling Study, Rome Air Development Center Technical Report, RADC-TR-76-247, 1976.

18. Gephart, L. S.; Greenwald, C. M.; Hoffman, M. M.; and Osterfeld, D. H., Software Reliability: Determination and Prediction, Air Force Flight Dynamics Laboratory Technical Report, AFFDL-TR-78-77, June 1978.

19. Mann, Nancy R; Schafer, Ray E.; and Singpurwalla, Nozer D., Methods for Statistical Analysis of Reliability and Life Data, John Wiley and Sons, New York, 1974.

20. Kline, M. B., "Software and Hardware R&M: What are the Differences," 1980 Proceedings of the Annual Reliability and Maintainability Symposium, 1980, pp. 179-185.

21. Bowen, D. C. and Shukal, J. M., "Software Reliability," RCA Engineer, Volume 25, No. 4, December 1979/January 1980, pp. 15-18.

22. Coutinho, J. de. S., "Software Reliability Growth," IEEE Symposium on Computer Software Reliability, 1973.

23. Wagoner, W., The Final Report on a Software Relibility Measurement Study, The Aerospace Corp., El Segundo, California, Report No. TOR-0074 (4112)-1, August 15, 1973.

24. Shooman, M., "Probabilistic Models for Software Reliability Prediction," International Symposium on Fault - Tolerant Computing, June 1972, IEEE Computer Society, New York.

25. Shooman, M., "Operational Testing and Software Reliability Estimation During Program Developments," Record of 1973 IEEE Symposium on Computer Software Reliability, IEEE Computer Society, New York, 1973.

26. Shooman, M., "Structural Models for Software Reliability Prediction," Proceedings of the 2nd International Conference on Software Engineering, October 1976, IEEE Computer Society, New York.

27. Shooman, M., "Spectre of Software Reliability and Its Exorcism," Proceedings of the Joint Automatic Control Conference, 1977, Published by the IEEE, New York, 1977, pp. 225-231.

28. Shooman, M. and Naturaja, S., Effect of Manpower Deployment and Bug Generation on Software Error Models, Rome Air Development Center Technical Report, RADC-TR-76-400, 1977.

29. Shooman, M., "Software Reliability," Computing Systems Reliability, 1979, pp 355-422.

30. Yau, Stephen and MacGregor, Terry, On Software Reliability Modeling, Rome Air Development Center Technical Report, RADC-TR-79-127, 1979.

31. Moranda, Paul L. and Jelinski, Z., Final Report on Software Reliability Study, McDonnell Douglas Astronautics Company, MDC Report No. 63921, 1972.

32. Tal, Jacob, Development and Evaluation of Software Reliability Estimators, University of Utah, Electrical Engineering Department, Salt Lake City, Utah, UTEC SR-77-013, 1976.

33. Schafer, R. E.; Alter, J. F.; Angus, J. E.; and Emoto, S. E., Validation of Software Reliability Models, Rome Air Development Center Technical Report, RADC-TR-79-147, 1979.

34. Littlewood, B. and Verrall, B. J., "A Bayesian Reliability Growth Model for Computer Software," The Journal of the Royal Statistical Society, Series C., Volume 22, No. 3, 1973, pp. 332-346.

35. Forman, Ernest H. and Singpurwalla, Nozer D., "Optimal Time Intervals for Testing Hypotheses on Computer Software Errors," IEEE Transactions on Reliability, Volume R-28, No. 3, 1979, pp. 250-253.

36. Jelinski, Z.; Moranda, P. B.; and Churchwell, J. B., Metrics of Software Quality - Final Technical Report, McDonnell Douglas Astronautics Company, Huntington Beach, Calif. 1980.

37. Lipow, M., "Models for Software Reliability," Proceedings of the Winter Meeting of the Aerospace Division of the American Society of Mechanical Engineers, 1978, 78-WA/Aero-18, pp. 1-11.

38. Lipow, M., "Some Variations of a Model for Software Time-to-Failure," TRW Systems Group, Correspondence ML-74-2260 1.9-21, August 1974.

39. Rushforth, C. K.; Staffanson, F. L.; and Crawford, A. E., Software Reliability Estimation Under Conditions of Incomplete Information, Rome Air Development Center Technical Report, RADC-TR-79-230, 1979.

40. Tal, J. and Barber, G. H., "Estimators For Software Reliability," Proceedings of the Automatic Control Conference, Volume 2, Publ[illegible] IEEE, New York, 1977, pp. 1067-1072.

41. Moranda, Paul, "Predictions of Software Reliability During Debugging," 1975 Proceedings of the Annual Reliability and Maintainability Symposium, Washington, D.C., 1975.

42. Moranda, Paul B., "Event-Altered Rate Models for General Reliability Analysis," IEEE Transactions on Reliability, Volume R-28, No. 5, 1979, pp. 376-381.

43. Schneidewind, Norman F., "Analysis of Error Processes in Computer Software," Sigplan Not., Volume 10, No. 6, 1975, pp. 337-346.

44. Goel, A. and Okumoto, K., "Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures," IEEE Transactions on Reliability, Volume R-28, No. 3, 1979, pp. 206-211.

45. Okumoto, Kazu and Goel, Amrit, "Optimum Release Time for Software Systems Based on Reliability and Cost Criteria," The Journal of Systems and Software, Volume 1, No. 4, 1980, pp. 315-318.

46. Sukert, Alan and Goel, Amrit, "A Guidebook for Software Reliability Assessment," 1980 Proceedings of the Annual Reliability and Maintainability Symposium, 1980, pp. 186-190.

47. Duane, J. T., "Learning Curve Approach to Reliability Monitoring," IEEE Transactions on Aerospace, Volume 2, 1964, pp. 563-566.

48. Evaluation Associates, Measurement and Forecast of Reliability Growth During Hardware Development, prepared for the Strategic Systems Project Office under contract number N00030-75-C-0048, CDRL 001, 1975.

49. Evaluation Associates, Modeling of SYS-1 Software Reliability and Its Growth, report number C11352, 1979.

50. Evaluation Associates, Reliability and Availability Evaluation Program Manual, 1981 Draft NAVSEA OD 29304B, prepared for the Department of the Navy, Strategic Systems Project Office, contract number N00030-79-C-0163 and N00123-79-D-0614M02.

51. Crow, Larry, Confidence Interval Procedures for Reliability Growth Analysis, Technical Report number 197, U.S. Army Material Systems Analysis Activity, Aberdeen Proving Grounds, Maryland, 1977.

52. Musa, John, "A Theory of Software Reliability and Its Applications," IEEE Transactions on Software Engineering, Volume SE-1, No. 3, 1975, pp. 312-327.

53. Musa, John, "Progress in Software Reliability Measurement," Second Software Life Cycle Management Workshop, Published by the IEEE, New York, 1978, pp. 153-155.

54. Musa, John, "Validity of Execution - Time Theory of Software Reliability," IEEE Transactions on Reliability, Volume R-28, No. 3, 1979, pp. 181-191.

55. Musa, John, "Software Reliability Measures Applied to System Engineering," Proceedings of the National Computer Conference, 1979, pp. 941-946.

56. Musa, John, "The Measurement and Management of Software Reliability," Proceedings of the IEEE, Volume 68, No. 9, 1980, pp. 1131-1143.

57. Chenoweth, H. B., "Modified Musa Theoretic Software Reliability," 1981 Proceedings of the Annual Reliability and Maintainability Symposium, 1981, pp. 353-356.

58. Musa, John, and Iannino, A. "Software Reliability Modeling - Accounting for Program Size Variation Due to Integration or Design Changes," ACM Sigmetrics Performance Evaluation Review, Volume 10, No. 2, pp. 16-25.

59. Brooks, W. D. and Motley, R. W., Analysis of Discrete Software Reliability Models, Rome Air Development Center Technical Report, RADC-TR-80-84, April 1980.

60. Littlewood, B., "A Bayesian Differential Debugging Model for Software Reliability," Proceedings IEEE Computer Society, International Computer Software Applications Conference, Published by IEEE, Piscataway, New Jersey, 1980, pp. 511-519.

61. Littlewood, Bev, "What Makes a Reliable Programs - Few Bugs, or a Small Failure Rate?" Proceedings of the National Computer Conference, 1980, pp. 707-713.

62. Littlewood, B., "Theories of Software Reliability: How Good Are They and How Can They Be Improved?" IEEE Transactions on Software Engineering, Volume 6, No. 5, 1980, pp. 489-500.

63. Littlewood, Bev, "Stochastic Reliability - Growth: A Model for Fault-Removal in Computer - Programs and Hardware - Designs," IEEE Transactions on Reliability, Volume R-30, No. 4, October 1981, pp. 313-320.

64. Littlewood, B., "A Critique of the Jelinski-Moranda Model for Software Reliability," 1981 Proceedings of the Annual Reliability and Maintainability Symposium, 1981, pp. 357-364.

65. Littlewood, B., "How to Measure Software Reliability and How Not To," IEEE Transactions on Reliability, Volume 28, No. 2, 1979, pp. 103-110.

66. Littlewood, B. and Sofer, A., "A Bayesian Modification to the Jelinski-Moranda Software Reliability Growth Model," unpublished report, Department of Operations Research, School of Engineering and Applied Science, George Washington Univerity.

67. Littlewood, B. and Verrall, J., "A Bayesian Reliability Growth Model for Computer Software," The Journal of the Royal Statistical Society, Series C, Volume 22, No. 3, 1973, pp. 332-346.

68. Littlewood, B., "Validation of a Software Model," Software Life Cycle Management Workshop, Atlanta, Georgia 1978, Published by the International Business Services, Inc., pp. 146-152.

69. Littlewood, B., "The Littlewood-Verrall Model for Software Reliability Compared with Some Rivals," J. System and Software, Volume 1, No. 3, 1980, pp. 251-258.

70. Thompson, W. E. and Chelson, P. O., "On the Specification and Testing of Software Reliability," 1980 Proceedings of the Annual Reliability and Maintainability Symposium, published by the IEEE, New York, pp. 379-383.

71. Haynes, Robert D. and Thompson, William E., "Hardware and Software Reliability and Confidence Limits for Computer-Controlled Systems," Microelectronics and Reliability, Volume 20, 1980, pp. 109-122.

72. Littlewood, Bev, "MTBF is Meaningless in Software Reliability," IEEE Transactions on Reliability, Volume 24, No. 1, 1975, p 82.

73. Trivedi, Ashok and Shooman, M., Computer Software Reliability: Many State Markov Modeling Techniques, Department of Electrical Engineering and Electrophysics, Polytechnic Institute of New York, New York, Technical Report POLY-EE/EP-75-005, 1975.

74. Littlewood, B., "Semi-Markov Model for Software Reliability with Failure Costs," Proceedings of the Symposium on Computer Software Engineering, New York, 1976, p. 281-300.

75. Littlewood, Bev, "Software Reliability Model for Modular Program Structure," IEEE Transactions on Reliability, Volume R-28, No. 3, 1979, pp. 241-246.

76. Sukert, Alan,, "An Investigation of Software Reliability Models," Proceedings 1977 Annual Reliability and Maintainability Symposium, 1977, pp. 478-484.

77. Sukert, Alan, "Analysis of Software Reliability Prediction Models," Avionics Reliability, its Techniques and Related Disciplines, AGARD Conference Proceedings No. 261, 1979, pp 34-1, 34-11.

78. Hughes Aircraft Company, Reliability Model Demonstration Study, Interim Technical Report, Fullerton, Calif., Contract No. F30602-80-C-0273, 1982.

79. Hansen, Gregory A., "Measuring Software Reliability," Mini-Micro Systems Volume 10, No. 8, August 1977, pp 54-57.

80. Wall, John K. and Ferguson, Paul A., "Pragmatic Software Reliability Prediction," Proceedings of the 1977 Annual Reliability and Maintainability Symposium, 1977, pp 485-488.

BIBLIOGRAPHY

Air Force Magazine, July 1973, pp. 46-57, Shelly, Martin, Computer Software Reliability Fact or Myth?, Ogden Air Logistics Center Technical Report, AD-780-697, 1973.

Anderson, H., Peiram, L.; and Strandberg, K., "A Study of Software Reliability," Ericsson Technics, No. 2, 1977, pp. 125-149.

Angus, J.; Schafer, R.; and Sukert, A., "Software Reliability Model Validation," 1980 Proceedings of the Annual Reliability and Maintainability Symposium, 1980, pp. 191-199.

Basin, S. L., "Estimation of Software Error Rates via Capture-Recapture Sampling," Science Applications, Inc., Palo Alto, California, 1973.

Basin, S. L., "Measuring the Error Content of Software," Science Applications, Inc., Palo Alto, California, 1974.

Bowen, D. C. and Shukal, J. N., "Software Reliability," RCA Engineer, Volume 25 Number 4, December 1979/January 1980, pp. 15-18.

Bowen, John, "Standard Error Classification to Support Software Reliability Assessment," 1980 National Computer Conference Proceedings, pp. 697-705.

Brown, J. R., and Lipow, M., "Testing for Software Reliability," Proceedings, 1975 International Conference on Reliable Software, IEEE Catalog Number 75 CH0940-7 CSR.

Brooks, W. D., and Motley, R. W., Analysis of Discrete Software Reliability Models, Rome Air Development Center Technical Report, RADC-TR-80-84, April 1980.

Chenoweth, H. B., "Modified Musa Theoretic Software Reliability", 1981 Proceedings of the Annual Reliability and Maintainability Symposium, 1981, pp. 353-356.

Corcoran, W. J.; Weingarten, H.; and Zehna, P., "Estimating Reliability After Corrective Action," Management Science, Volume 10, Number 4, 1954, pp. 786-795.

Coutinho, J.de.S., "Software Reliability Growth," IEEE Symposium on Computer Software Reliability, 1973.

Crow, Larry, Confidence Interval Procedures for Reliability Growth Analysis, Technical Report #197, U.S. Army Material Systems Analysis Activity, Aberdeen Proving Grounds, Maryland, 1977.

BIBLIOGRAPHY (Cont'd)

Duane, J. T., "Learning Curve Approach to Reliability Monitoring," IEEE Transactions on Aerospace, Volume 2, 1964, pp. 563-566.

Duvall, L.; Martens, J.; Swearingen, D.; and Donahoo, J., "Data Needs for Software Reliability Modeling," 1980 Proceedings of the Annual Reliability and Maintainability Symposium, 1980, pp. 200-208.

Elliott, R. W.; Marchbank, M. P.; McWilliams, M. G.; Ringer, L. J.; and Simmons, D. B.; "Measuring Computer Software Reliability," Computer and Industrial Enginneering, Volume 2, Number 3, 1978, pp. 141-151.

Evaluation Associates, Measurement and Forecast of Reliability Growth During Hardware Development, prepared for the Strategic Systems Project Office under contract # N00030-75-C-0048, CRDL 001, 1975.

Evaluation Associates, Modeling of SYS-1 Software Reliability and Its Growth, Report # C1135-2, 1979.

Evaluation Associates, Reliability and Availability Evaluation Program Manual, 1981 Draft NAVSEA OD 29304B, prepared for the Department of the Navy, Strategic Systems Project Office, Contract # N00030-79-C-0163 and N00123-79-D-06164M02.

Forman, Ernest H.; and Singpurwalla, Nozer D.; "Optimal Time Intervals for Testing Hypotheses on Computer Software Errors," IEEE Transactions on Reliability, Volume R-28, Number 3, 1979, pp. 250-253.

Gephart, L.S., Greenwald, C.M.; Hoffman, M.M.; and Osterfeld, D.H.; Software Reliability: Determination and Prediction, Air Force Flight Dynamics Laboratory Technical Report, AFFDL-TR-78-77, June 1978.

Gilb, Tom, Software Metrics, Winthrop Publishers, Inc., Cambridge, Massachusettes, 1977.

Gilb, Tom, "Sofware Metrics - The Emerging Technology," Data Management, Volume 13, Number 7, 1975, pp. 34-37.

Gilb, Tom, "The Measurement of Software Reliability and Maintainability: Some Unconventional Approaches to Reliable Software," Computers and People, Volume 26, N9, 1977, pp. 16-21.

Goel, A. and Okumoto, K., "Time-Dependent Error-Detection Rate for Software Reliability and Other Performance Measures," IEEE Transactions on Reliability, Volume R-28, Number 3, 1979, pp. 206-211.

Hansen, Gregory A., "Measuring Software Reliability," Mini-Micro Systems, Volume 10, Number 8, August 1977, pp. 54-57.

BIBLIOGRAPHY (Cont'd)

Haynes, Robert D. and Thompson, William E., "Hardware and Software Reliability and Confidence Limits for Computer-Controlled Systems," Microelectronics and Reliability, Volume 20, 1980, pp. 109-122.

Hecht, Herbert, Measurement, Estimation, and Prediction of Software Reliability, National Aeronautics and Space Administration Technical Report, NASA-CR-145135, prepared for the Aerospace Corporation, El Segundo, California, 1977.

Hecht, H.; Sturm, W.; and Trattner, S., "Reliability Measurement During Software Development," Proceedings of Computers in Aerospace Conference, AIAA, 1977, pp. 404-412.

Heiner, Guenter, "Introduction to Software Reliability - A Key Issue of Computing Systems Reliability," Avionics Reliability, its Techniques and Related Disciplines, AGARD Conference Proceedings Number 261, 1979, pp. 30.1-30.13.

Hughes Aircraft Company, Reliability Model Demonstration Study, Interim Technical Report, Fullerton, California, Contract N. F30602-80-C-0273, 1982.

Jelinski, Z.; Moranda, P. B.; and Churchwell, J. B., Metrics of Software Quality-Final Technical Report, McDonnell Douglas Astronautics Company, Huntington Beach, California, 1980.

Kline, M.B., "Software and Hardware R&M: What are the Differences," 1980 Proceedings of the Annual Reliability and Maintainability Symposium, 1980, pp. 179-185.

LaPadula, L.J., Engineering of Quality Software Systems, Volume VIII - Software Reliability Modeling and Measurement Techniques, Mitre Corporation, Rome Air Development Center Technical Report, RADC-TR-74-325, Volume III, January 1975.

Lipow, M., "Some Variations of a Model for Software Time-to-Failure," TRW Systems Group, Correspondence ML-74-2260 1.9-21, August 1974.

Littlewood, B. and Verrall, J., "A Bayesian Reliability Growth Model for Computer Software," The Journal of the Royal Statistical Society, Series C, Volume 22, Number 3, 1973, pp. 332-346.

Littlewood, Bev, "MTBF is Meaningless is Software Reliability," correspondence, IEEE Transactions on Reliability, Volume 24, Number 1, 1975, p. 82.

Littlewood, B., "Semi-Markov Model for Software Reliability with Failure Costs," Proceedings of the Symposium on Computer Software Engineering, New York, 1976, pp. 281-300.

BIBLIOGRAPHY (Cont'd)

Littlewood, B., "Validation of a Software Model," Software Life Cycle Management Workshop, Atlanta, Georgia, 1978, Published by the International Business Services, Inc., pp. 146-152.

Littlewood, B., "How to Measure Software Reliability and How Not To," IEEE Transactions on Reliability, Volume 28, Number 2, 1979, pp. 103-110.

Littlewood, Bev, "Software Reliability Model for Modular Program Structure," IEEE Transactions on Reliability, Volume R-28, Number 3, 1979, pp. 241-246.

Littlewood, Bev, "What Makes a Reliable Program - Few Bugs, or a Small Failure Rate?," Proceedings of the National Computer Conference, 1980, pp. 707-713.

Littlewood, B., "A Bayesian Differential Debugging Model for Software Reliability," Proceedings IEEE Computer Society, International Computer Software Applications Conference, Published by IEEE, Piscataway, New Jersey, 1980, pp. 511-519.

Littlewood, B., "The Littlewood-Verrall Model for Software Reliability Compared with Some Rivals," J. System and Software, Volume 1, Number 3, 1980, pp. 251-258.

Littlewood, B., "Theories of Software Reliability: How Good Are They and How Can They Be Improved?," IEEE Transportation on Software Engineering, Volume 6, Number 5, 1980, pp. 489-500.

Littlewood, Bev, "Stochastic Reliability-Growth: A Model for Fault-Removal in Computer-Programs and Hardware-Designs," IEEE Transactions on Reliability, Volume R-30, Number 4, October 1981, pp. 313-320.

Littlewood, B., "A Critique of the Jelinski-Moranda Model for Software Reliability," 1981 Proceedings of the Annual Reliability and Maintainability Symposium, 1981, pp. 357-364.

Littlewood, B. and Sofer, A., "A Bayesian Modification to the Jelinski-Moranda Software Reliability Growth Model," unpublished report, Department of Operations Research, School of Engineering and Applied Science, George Washington University.

Lipow, M., "Models for Software Reliability," Proceedings of the Winter Meeting of the Aerospace Division of the American Society of Mechanical Engineers, 1978, 78-WA/Aero-18,pp. 1-11.

Mann, Nancy R.; Schafer, Ray E.; and Singpurwalla, Nozer D., Methods For Statistical Analysis of Reliability and Life Data, John Wiley & Sons, New York, 1974.

BIBLIOGRAPHY (Cont'd)

Maxwell, F. D., The Determination of Measures of Software Reliability, National Aeronautics and Space Administration Technical Report, NASA-CR-158960, prepared by the Aerospace Corporation, El Segundo, California, 1978.

Mills, H. D., On The Statistical Validation of Computer Programs, FSC-72-6015, IBM Federal Systems Division, Gaithersberg, Maryland, 1972.

Miyamoto, Isao, "Software Reliability in Online Real Time Environment," Proceedings of the International Conference on Reliable Software, Published by the IEEE, New York, 1975, pp. 194-203.

Miyamoto, Isao, "Reliability Evaluation and Management for an Entire Software Life Cycle," Second Software Life Cycle Management Workshop, 1978, Published by the IEEE, New York, pp. 195-208.

Moranda, Paul L. and Jelinski, Z., Final Report on Software Reliability Study, McDonnell Douglas Astronautics Company, MDC Report Number 63921, 1972.

Moranda, Paul B., "Software Reliability Predictions," Proceedings of the Sixth Triennial World Congress of the International Federation of Automatic Control, 1975, pp. 34.2-34.7.

Moranda, P.B., "The (Sad) Status of; (Unapproved) Limits To; and (Manifold) Alternatives for Software Measurement Techniques," Proceedings of the Spring Compcon 1978, Published by the IEEE, New York, 1978, pp. 353-354.

Mirjamoto, Isao, "Reliability Evaluation and Management for an Entire Software Life Cycle," Second Software Life Cycle Management Workshop, 1978, Published by the IEEE, New York, pp. 195-208.

Moranda, Paul, "Predictions of Software Reliability During Debugging," 1975 Proceedings of the Annual Reliability and Maintainability Symposium, Washington, D.C., 1975.

Moranda, Paul B., "Event-Altered Rate Models for General Reliability Analysis," IEEE Transactions on Reliability, Volume R-28, Number 5, 1979, pp. 376-381.

Musa, John, "A Theory of Software Reliability and Its Application," IEEE Transactions on Software Engineering, Volume SE-1, Number 3, 1975, pp. 312-327.

Musa, John, "Progress in Software Reliability Measurement," Second Software Life Cycle Management Workshop, Published by the IEEE, New York, 1978, pp. 153-155.

Musa, John, "Validity of Execution-Time Theory of Software Reliability," IEEE Transactions on Reliability, Volume R-28, Number 3, 1979, pp. 181-191.

Musa, John, "Software Reliability Measures Applied to System Engineering," Proceedings of the National Computer Conference, 1979, pp. 941-946.

BIBLIOGRAPHY (Cont'd)

Musa, John, "The Measurement and Management of Software Reliability," Proceedings of the IEEE, Volume 68, Number 9, 1980, pp. 1131-1143.

Musa, J., "Program for Software Reliability and System Test Schedule Estimation - User's Guide," IEEE Computer Society Repository, Reference Number R77-244, p. B-4.

Musa, John and Iannino, A., Software Reliability Modeling - Accounting for Program Size Variation Due to Integration or Design Changes, ACM Sigmetrics Performance Evaluation Review, Volume 10, Number 2, pp. 16-25.

Myers, Glenford, J., Software Reliability Principles and Practices, John Wiley and Sons, New York, 1976.

Neighbors, M. A., "Assuring Software Reliability," Computer Decisions, Volume 8, Number 12, 1976, pp. 44-46.

Nelson, Eldred, "Estimating Software Reliability From Test Data," Microelectronics and Reliability, Volume 17, Number 1, 1978, pp. 67-73.

Okumoto, Kazu and Goel, Amrit, "Optimum Release Time for Software Systems Based on Reliability and Cost Criteria," The Journal of Systems and Software, Volume 1, Number 4, 1980, pp. 315-318.

Parr, F. N., and Lehman, M. M., State of the Art Survey of Software Reliability, Publication 77/15, Department of Computing and Control, Imperial College, London, England, May 1977.

Ramamoorthy, C. V., and Bastani, F. B., "Modeling of the Software Reliability Growth Process," Proceedings of the IEEE Computer Society, International Computer Software Applications Conference, Published by IEEE, Piscataway, New Jersey, 1980, pp. 161-169.

Ramamoorthy, C. V., and Bastani, F. B., "Software Reliability - Status and Perspectives", IEEE Transactions on Software Engineering, Volume SE-8, Number 4, July 1982, pp. 354-371.

Rault, J. C., Memmi, G., and Pimont, S., "Quantitative Assessments of Software Reliability," Avionics Reliability, its Techniques and Related Disciplines, AGARD Conference Proceedings, Number 261, 1979, pp. 33.1-33.11.

Rudner, B., "Seeding/Tagging Estimations of Software Errors: Models and Estimates," RADC-TR-77-15, Rome Air Development Center, A036655, 1977.

Rushforth, C. K.; Staffanson, F. L.; and Crawford, A.E., Software Reliability Estimation Under Conditions of Incomplete Information, Rome Air Development Center Technical Report, RADC-TR-79-230, 1979.

BIBLIOGRAPHY (Cont'd)

Schafer, R. E.; Alter, J. F.; Angus, J. E.; and Emoto, S. E., Validation of Software Reliability Models, Rome Air Development Center Technical Report, RADC-TR-79-147, 1979.

Schick, George J. and Wolverton, Ray W., "An Analysis of Competing Software Reliability Models," IEEE Transactions on Software Engineering, Volume SE-4, Number 2, 1978, pp. 104-120.

Schneidewind, Norman, A Methodology for Software Reliability Prediction and Quality Control, Technical Report # NPS-55SS72111A, Naval Post Graduate School, Monterey, California, 1972.

Schneidewind, Norman F., "Analysis of Error Processes in Computer Software," Sigplan Not., Volume 10, Number 6, 1975, pp. 337-346.

Shooman, M., "Operational Testing and Software Reliability Estimation During Program Developments," Record of 1973 IEEE Symposium on Computer Software Reliability, IEEE Computer Society, New York, 1973.

Shooman, M., "Probabilistic Models for Software Reliability Prediction," International Symposium on Fault-Tolerant Computing, June 1972, IEEE Computer Society, New York.

Shooman, M., "Software Reliabiity: Measurement and Models," Proceedings of the 1975 Annual Reliability and Maintainability Symposium, 1975, pp. 485-491.

Shooman, M., "Structural Models for Software Reliability Prediction," Proceedings of the 2nd International Conference on Software Engineering, October 1976, IEEE Computer Society, New York.

Shooman, M., "Software Reliability: Analysis and Prediction," Integrity in Electronic Flight Control Systems, AGARDograph Number 224, Advisory Group for Aerospace Research and Development, Part II, p. 7, 1977.

Shooman, N., "Spectre of Software Reliability and Its Exorcism," Proceedings of the Joint Automatic Control Conference, 1977, Published by the IEEE, New York, 1977, Volume 1, pp. 225-231.

Shooman, M. and Naturaja, S., Effect of Manpower Deployment and Bug Generation on Software Error Models, Rome Air Development Center Technical Report, RADC-TR-76-400, 1977.

Shooman, M., "Software Reliability," Computing Systems Reliability, 1979, pp. 355-422.

Soi, I. M. and Gopal, K., "A Comparative Study of Software Reliability Models," Journal of the Institute of Electron Telecommunications Eng. Division, Volume 59, NET-1, pp. 1-6.

BIBLIOGRAPHY (Cont'd)

Suguira, N.; Yamamoto, M.; and Shiino, T., "On the Software Reliability," Microelectronics and Reliability, Volume 13, 1974, pp. 529-533.

Sukert, Alan, A Software Reliability Modeling Study, Rome Air Development Center Technical Report, RADC-TR-76-247, 1976.

Sukert, Alan, "An Investigation of Software Reliability Models," Proceedings 1977 Annual Reliability and Maintainability Symposium, 1977, pp. 478-484.

Sukert, Alan, "Analysis of Software Reliability Prediction Models," Avionics Reliability, its Techniques and Related Disciplines, AGARD Conference Proceedings Number 261, 1979, pp. 34-1, 34-11.

Sukert, Alan and Goel, Amrit, "A Guidebook for Software Reliability Assessment," 1980 Proceedings of the Annual Reliability and Maintainability Symposium, 1980, pp. 186-190.

Tal, Jacob, Development and Evaluation of Software Reliability Estimators, University of Utah, Electrical Engineering Department, Salt Lake City, Utah, UTEC SR77-013, 1976.

Tal, Jacob; Barber, G. H.; and Timothy, L. K., "Development and Evaluation of Software Reliability Estimators," Proceedings Hawaii International Conference of Systems Science 10th, Published by West Period Company, California, 1977, pp. 230-233.

Tal, Jacob, Development and Evaluation of Software Reliability Estimators, Electrical Engineering Department Technical Report, University of Utah, UTEC-SR-77-013, 1976.

Tal, J. and Barber, G. H., "Estimators For Software Reliability," Proceedings of the Jt. Automatic Control Conference, Volume 2, Published by IEEE, New York, 1977, pp. 1067-1072.

Thayer, T. A.; Craig, G. R.; and Frey, L. E., Software Reliability Study, Rome Air Development Center Technical Reprot, RADC-TR-76-238, 1976.

Thayer, T. A.; Craig, G. R.; Fray, L. E.; Hetrick, W. L.; and Lipow, M., Software Reliability Study, Rome Air Development Center Technical Report, RADC-TR-76-238, 1976.

Thompson, W. E. and Chelson, P. O., "On the Specification and Testing of Software Reliability," 1980 Proceedings of the Annual Reliability and Maintainability Symposium, published by the IEEE, New York, pp. 379-383.

BIBLIOGRAPHY (Cont'd)

Trivedi, Ashok and Shooman, M., Computer Software Reliability: Many State Markov Modeling Techniques, Department of Electrical Engineering and Electrophysics, Polytechnic Institute of New York, New York, Technical Report POLY-EE/EP-75-005, 1975.

Wagoner, W., The Final Report on a Software Reliability Measurement Study, The Aerospace Corporation, El Segundo, California, Report No. TOR-0074 (4112)-1, August 15, 1973.

Wall, John K. and Ferguson, Paul A., "Pragmatic Software Reliability Prediction," Proceedings of the 1977 Annual Reliability and Maintainability Symposium, 1977, pp. 485-488.

Yau, Stephen and MacGregor, Terry, On Software Reliability Modeling, Rome Air Development Center Technical Report, RADC-TR-79-127, 1979.

DISTRIBUTION

	Copies
Defense Printing Service Washington Navy Yard Washington, DC 20374	1
Naval Air Development Center Warminster, PA 18974 Attn: Technical Library Code 5033 (H. Stuebing)	1
Naval Ocean Systems Center 271 Catalina Boulevard San Diego, CA 92152 Attn: Code 9134 (R. Crabb) Attn: Code 9133 (L. McCoy)	1 1
Naval Postgraduate School Monterey, CA 93940 Attn: Code 52 CL (L. Cox)	1
Naval Research Laboratory Washington, DC 20375 Attn: Code 7503 (K. Heninger)	1
Naval Ship Research & Development Center Bethesda, MD 20084 Attn: Technical Library Code 1828 (M. Culpepper)	1
Naval Training Equipment Center Orlando, FL 32813 Attn: Technical Library Code N-74	1
Naval Underwater Systems Center Newport, RI 02840 Attn: Code 4451 (R. Kasik)	1
USN Oceanographic Office NSTL Station Bay St. Louis, MI 39522 Attn: Code 9220, Mr. Douglas S. Gordon	1
Naval Weapons Center China Lake, CA 93555 Attn: Code 31302 (J. Zenor)	1
Navy Personnel Research & Development Center San Diego, CA 92152 Attn: Code P204 (M. Underwood)	1
FCDSSA, Dam Neck Virginia Beach, VA 23461 Attn: Code 00T	1
GIDEP Operations Office Corona, CA 91720	1
Library of Congress Attn: Gift and Exchange Division Washington, DC 20540	4
Dr. Robert B. Grafton Office of Naval Research Code 411IS 800 N. Quincy Street Arlington, VA 22217	1
Mr. Frank E. McGarry Code 582.1 NASA/GSFC Greenbelt, MD 20771	1

DISTRIBUTION (Cont.)

	Copies
Mr. Ray Paul NAVSEA Code 06C31 Reliability, Maintability, Quality Assurance Bldg. NC2 Crystal City, VA 22202	1
Dr. Larry Crow US Army Material Systems Analysis Activity Aberdeen Proving Ground, MD 21005	1
Charles Stark Draper Laboratory 68 Albany Street Cambridge, MA 02139 Attn: Mail Station 33 Division 40-B	2
Compro, Incorporated 24 Marshall Place Fredericksburg, VA 22401	1
SDC Integrated Services, Incorporated 601 Caroline Street Fredericksburg, VA 22401	1
DACS RADC/ISISI Griffiss AFB, NY 13441	1
General Electric Company Ordnance Systems Electronic Systems Division 100 Plastics Avenue Pittsfield, MA 01201 Attn: Code ASA Code WCCAE	2 1
GSG, Incorporated 51 Main Street Salem, NH 03079 Attn: Mr. Frank Hill	1

	Copies
EG&G, Washington Analytical Services Center, Inc. Attn: Library Oliver Smith P.O. Box 552 Dahlgren, VA 22448	1 1
Sperry Univac, Incorporated Dahlgren, VA 22448	1
Mr. James Dobbins IBM MS 105/013 9500 Godwin Drive Manassas, Virginia 22110	1
Mr. Ted Workman Hewlett Packard Computer Systems Division 19447 Pruneridge Avenue Cupertino, California 95014	1
Mr. Stuart Glickman Norden Systems Box 5300 Norwalk, Connecticut 06856	1
Mr. Roger Martin National Bureau of Standards Institute for Computer Science and Technology Room B-266, Bldg. 225 Washington, D.C. 20234	1
Mr. Walter Ellis Advisory Systems Analyst IBM Federal Systems Division 6600 Rockledge Drive Bethesda, Maryland 20034	1
Mr. Frank Salvia Systems Management Sperry Corporation Electronic Systems MS - M5 Great Neck, New York 11020	1

DISTRIBUTION (Cont.)

	Copies
Laura L. Good Development Systems Operation INTEL Corporation 3065 Bowers Avenue Santa Clara, California 95051	1
Mr. John Katuzny Raytheon Inc., Sub Sig. Division Portsmonth, R.I. 02871	
G.W. Ferreira G.E. Ordnance Systems #OP 9 100 Plastics Ave. Pittsfield, Mass. 01201	1
Dr. John D. Muse Supervisor, Computer Measurements, Security, and Software Reliability Bell Laboratories Whippany Road Whippany, NJ	1
Professor Amrit L. Goel Department of Industrial Engineering Syracuse University Syracuse, NY 13210	2
RADC/COEE Rome Air Development Center Griffiss AFB, NY 13441	2
Professor Norman Schneidewind Chairman Code 54Ss Naval Postgraduate School Monterey, CA 92940	1
Dr. Victor Basili University of Maryland Dept. of Computer Sciences College Park, MD 20742	1
Mr. Robert A. Clarke Space Missions & Engineering Services Operations 10440 State Highway 83 Colorado Springs, CO 80908	1
Mr. J. E. Angus Hughes Aircraft Company P.O. Box 3310 Fullerton CA 92634	1
W. D. Brooks/R. W. Motley IBM Corporation/Federal Systems Division 18100 Frederick Pike Gaithersburg, MD 20760	1
Mr. William Hatch Automated Sciences Group, Inc. Rt. 2 Box 261 King George, VA 22485	1
Professor John Reynolds Dept. of Computer Science Mary Washington College Fredericksburg, VA 22401	1
Internal Distribution:	
E35	1
E431	10
E432	2
F02	1
F10	1
F16	1
F16 (Lemoine)	1
F20	1
F22	1
F24	1
F26	1
F28	1
F42	1
F42 (Spooner)	1
F44	1
F50	1
F56 (Prehoda)	1
G10	1

DISTRIBUTION (Cont.)

	Copies
G11	1
G12 (Batayte)	1
G13 (Moore)	1
G21 (Clawson)	1
G42	1
K02	1
K04 (Warner)	1
K05	1
K10	1
K105 (Gemmill)	1
K106 (Crigler)	1
(Thomas)	1
K11	1
K12	1
K13	1
K14	1
K14 (Clark)	1
K20	1
K30	1
K301	1
K304	1
K33	1
K34	1
K35	1
K40	1
K402	1
K41	1
K42	1
K43	1
K44	1
K50	1
K502	1
K51	2
K52	1
K52 (Farr)	15
(J. Smith)	1
K53	1
K54	1
K54 (McCoy)	1
N14	1
N20	1
N20A	1
N21 (McConnell)	1
N22	1
N31	1
N32	1
N43	1
N51	1
N52	1
N53	1
R44	1
U02	1
U12	1
U22	1
U23	1
U30	1
U31	1
U32	1